UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE MATEMÁTICAS

Departamento de Sistemas Informáticos y Programación

TÉCNICAS COALGEBRAICAS Y CATEGÓRICAS PARA EL ESTUDIO DE LAS SEMÁNTICAS DE PROCESOS

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Ignacio Fábregas Alfaro

Bajo la dirección de los doctores

David de Frutos Escrig
Miguel Palomino Tarjuelo

Madrid, 2012

Técnicas Coalgebraicas y Categóricas para el Estudio de las Semánticas de Procesos

TESIS DOCTORAL

Ignacio Fábregas Alfaro

Programa de Doctorado de Ingeniería Informática
Departamento de Sistemas Informáticos y Programación
Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid

2011

Técnicas Coalgebraicas y Categóricas para el Estudio de las Semánticas de Procesos

Tesis doctoral dirigida por los doctores

David de Frutos Escrig

Miguel Palomino Tarjuelo

Programa de Doctorado de Ingeniería Informática

Departamento de Sistemas Informáticos y Programación

Facultad de Ciencias Matemáticas

Universidad Complutense de Madrid

2011

Resumen

De un modo general, esta tesis puede englobarse dentro del estudio de los sistemas de procesos y, muy especialmente, del estudio de las relaciones entre esos procesos mediante bisimulación y simulación (abreviadamente (bi)simulación). Además, los resultados que aquí aparecen están sustentados en gran parte por el marco general de la *teoría de categorías* y, más concretamente, en el de las *coálgebras* en el que Jesse Hughes y Bart Jacobs propusieron una definición general de simulación entre coálgebras. La noción de simulación entre coálgebras se apoya en los conceptos de orden funtorial y alzamiento de relaciones; la bisimulación entre coálgebras es un caso particular de simulación (en la que el orden funtorial es la igualdad).

La (bi)simulación coalgebraica permite disponer de un único concepto general cuyas instancias definen las nociones concretas para sistemas de transiciones (LTS), estructuras de Kripke, etc. No obstante no todo orden funtorial define una relación de similitud correcta pues, en concreto, es necesario que se cumpla la condición de *estabilidad* para asegurar que la composición de simulaciones es también una simulación, y así garantizar que la similitud sea una relación transitiva.

Podemos considerar dos partes esenciales en esta tesis: en la primera explotamos la potencia de la definición coalgebraica de simulación para obtener resultados generales, mientras que en la segunda nos centramos en el estudio de dos nuevas nociones de simulación que hemos propuesto, pero en esta ocasión desde un punto de vista más clásico al estilo de los resultados de van Glabbeek.

En primer lugar, estudiamos bajo qué condiciones las instancias de dichas nociones categóricas generales de (bi)simulación permiten traspasar propiedades entre las estructuras relacionadas, es decir, en qué condiciones se puede generalizar al mundo coalgebraico los resultados clásicos de reflexión y preservación de propiedades lógicas mediante bisimulaciones, y la preservación mediante simulaciones. Nuestra conclusión es que, si bien para el caso de las bisimulaciones se obtiene la deseada generalización, para el caso de la simulación es necesario restringir los órdenes que participan en la definición de simulación propuesta de Hughes y Jacobs, obteniéndose en este caso resultados parciales.

También estudiamos cómo las transformaciones naturales entre funtores permiten unificar nociones de (bi)simulación entre distintos tipos de estructuras, obteniendo resultados de representación de las coálgebras entre los distintos funtores relacionados por la transformación natural. En particular, unificamos los sistemas probabilísticos y los sistemas de transiciones en el modelo de los sistemas de multitransiciones.

Para finalizar la primera parte de la tesis, profundizamos en el estudio de la no-

ción de simulación coalgebraica y de la condición necesaria de estabilidad, para llegar a entender cómo se pueden definir nociones de simulación dentro del marco coalgebraico que definan relaciones de similitud con buenas propiedades. En concreto, ilustramos nuestros resultados teóricos definiendo dos nuevas nociones de simulación sobre sistemas de transiciones etiquetados: la *simulación covariante-contravariante* (cc-simulación) y la *simulación conforme*.

La segunda parte de la tesis está dedicada al estudio de las dos nociones de simulación que hemos nombrado. Para las dos nociones de simulación construimos su caracterización lógica y axiomática. Además, para el caso de la simulación covariante-contravariante, estudiamos también su estrecha relación con los *refinamientos modales* sobre sistemas de transiciones modales (MTS) construyendo transformaciones de los LTS módulo cc-simulación y los MTS módulo refinamiento modal, y viceversa. Asimismo, construimos las fórmulas características de los LTS módulo cc-simulación y cómo estas representan procesos, es decir, la *representación gráfica* de las fórmulas.

Agradecimientos

Reconozco que yo no estaría aquí si no fuera por la ayuda (directa e indirecta) de una buena cantidad de personas. Imagino que habrá quien sea capaz de escribir una tesis doctoral sin demasiada ayuda, pero ni ese es mi caso ni tampoco creo que sea el de la mayoría de los investigadores. Por ello, me resulta muy gratificante escribir esta sección, que si bien no tiene carga científica (¡ya hay ciencia suficiente en el resto de hojas!), no por eso carece de interés, pues es el lugar indicado para nombrar a las personas que me han permitido llegar hasta aquí.

Estando en tercero de la Licenciatura de Matemáticas tenía que decidir qué especialidad elegiría para cursar durante los siguientes dos años. Mis preferencias personales eran computación, fundamental e, incluso, astronomía. En el fondo estuve ante la misma situación cuando empecé en la universidad y dudaba estudiar entre matemáticas, física o informática. En la licenciatura descarté enseguida la opción de la astronomía, que era la que menos me interesaba, acotando la elección entre el perfil de computación o el de fundamental. Esa elección la resolví al considerar que ya que había escogido estudiar matemáticas dejando de lado a la informática, merecía la pena estudiar la especialidad de computación. Para hacer la elección me ayudó que el primer año de la carrera la asignatura de Informática fue la que mejor se me dio.

Durante las asignaturas del perfil conocí a muchos de los profesores a los que hoy quiero agradecer. Por eso mismo quiero empezar por David de Frutos. A David lo conocí el último año de carrera y por partida doble. Era mi profesor en la asignatura del perfil "teoría de la programación", pero fue mucho más importante para mí el que fuera también mi tutor ese año en un "trabajo académicamente dirigido". En la hora semanal que David nos dedicó a otro compañero y a mí, estudiamos los libros clásicos de Hoare y Milner, pilares de la teoría de la concurrencia y de mi investigación. Aparte del propio temario que nos enseñó David, a lo largo de ese año descubrí dos cosas sobre él: en las clases de David se pensaba hasta el ejercicio con apariencia más inofensiva... que, además, resultaba no ser en absoluto tan "inocente"; y que es difícil encontrarlo en su despacho. Ya una vez que David se convirtió en co-director de esta tesis, comprendí que no era únicamente que David estuviese en multitud de reuniones, sino que gran parte del tiempo estaba en el despacho de alguno de sus doctorandos.

A David no solo le estoy profundamente agradecido por descubrirme a "los clásicos" de la materia, sino especialmente por la ingente cantidad de horas que me ha dedicado (¡muchos alumnos suyos le habrán buscado en su despacho cuando estaba en el mío!). Han sido muchísimas las discusiones científicas que hemos tenido y muchas las horas que

he dedicado con David a entender un artículo o, sencillamente, algún concepto que se me escapaba. Es más, no han sido pocos los días en los que ambos hemos salido de la facultad una vez que esta había cerrado e, incluso, los días festivos que nos hemos visto para terminar algún artículo a tiempo.

A mi otro co-director, Miguel Palomino, no le conocí durante la licenciatura, sino después de hacer los cursos de doctorado. Era mi segundo año de estudios en el programa de doctorado, el año en el que me tocaba hacer el trabajo de investigación. Miguel venía para echarnos una mano con la teoría de categorías y, claramente, sin su ayuda tampoco hubiese sido posible haber llegado hasta aquí. Gracias a Miguel he conseguido entender parte del intrincado mundo que es la teoría de categorías. Aún recuerdo mi desconcierto inicial, leyendo artículos de Bart Jacobs, al tratar de entender qué se suponía que determinado diagrama conmutativo demostraba; pero allí estaba Miguel no solo para explicármelo sino para animarme recordándome que, en efecto, no era nada trivial comprobar que dicho diagrama funcionaba realmente, por lo que no era de extrañar que llevase dos páginas "poniendo en palabras" el citado diagrama.

Al igual que David, Miguel siempre ha estado disponible para atender mis consultas, ayudarme y discutir cualquier cosa. De hecho, ambos se han complementado a la perfección y sé que sin la paciencia y ayuda de ambos, ahora mismo no habría ninguna tesis que leer.

Quiero agradecer también de manera muy especial a los otros dos co-autores de mis publicaciones: Luca Aceto y Anna Ingólfsdóttir. A ambos los conocí en Reykjavik gracias al programa NILS y sus becas Abel. Allí pasé un mes junto con Miguel. Ninguno de los dos conocíamos personalmente a Luca, pero David sí y, de hecho, unos meses antes de nuestra estancia ya David había pasado cuatro meses en Reykjavik junto con Carlos Gregorio, trabajando también con Luca. Después de mi estancia, vinieron Luca y Anna a Madrid a continuar la colaboración. Considero que esta colaboración ha sido muy positiva (y sinceramente espero que lo continúe siendo) tanto científica como personalmente, pues no solo considero que es siempre interesante conocer a otros investigadores y aprender de su experiencia, sino que en el caso particular de Luca y Anna ha sido un placer poder compartir tiempo juntos.

Quiero mostrar también mi agradecimiento a Narciso Martí, no ya solo por su amplia revisión de esta tesis (que sin duda se ha visto mejorada notablemente tras pasar por sus manos), sino por la detallada y completa revisión que ya realizó hace unos años para mi trabajo de tercer ciclo. Asimismo, debido a la naturaleza de esta tesis que se presenta como colección de artículos no quiero olvidarme de todos los revisores anónimos que con su trabajo y comentarios contribuyeron en su momento a que las publicaciones que aparecen aquí se viesen muy mejoradas.

Le debo mucho a muchos compañeros de departamento que además empezaron siendo mis profesores. De entre ellos quiero empezar agradeciendo a Yolanda Ortega, Jesús Escribano, Mª Inés Fernández y Jorge Carmona, con los que tras el paso del tiempo pasaron de darme clase (aunque en el caso de Jorge nunca llegó a ser mi profesor) a compartir comidas y agradables charlas en la facultad. Reservo un hueco especial para los compañeros Luis Llana, Carlos Gregorio y César Andrés: a Luis le estaré eternamente agradecido por sus innumerables ayudas con Linux y LaTeX, a Carlos le debo en concreto que me haya ayudado con el formato de mi tesis y a César su impagable trabajo con el

servidor `simba`. También quiero aprovechar esta ocasión para nombrar a Manuel Núñez y Mercedes G. Merayo, con los que he compartido más de un congreso y muchas charlas.

Por supuesto, tampoco quiero olvidarme de los compañeros del despacho de al lado: Lidia Sánchez Gil y David Romero. A Lidia la conozco desde hace bastante tiempo (es sorprendente que no coincidiéramos en alguna clase de la licenciatura) durante el que hemos pasado muy buenos momentos, en especial en Marktoberdorf. Además quiero agradecer a Lidia que se haya ofrecido a repasar la tesis, con todo lo que eso conlleva. A ambos les agradezco esas charlas "respiro" en mitad de la mañana que necesita cualquier persona para desconectar de la investigación. Al hilo del respiro, también quiero hacer una mención honorífica a todos mis amigos y amigas que, aunque no han podido ayudarme en mi investigación de manera directa, me han proporcionado esas horas de relax que han convertido este viaje en algo mucho más agradable.

Finalmente, no me olvido de mis padres. En primer lugar por lo obvio: no estaría en el mundo sin ellos; en segundo por lo no tan obvio pero igualmente importante: siempre me han apoyado y aguantado incluso durante esas semanas más estresantes que, necesariamente, aparecen mientras se escribe una tesis.

Índice general

Capítulo 1

Introdución

El presente capítulo trata de dar una visión global de la situación actual de la investigación en el campo en el que se enmarca esta tesis, así como su evolución a lo largo de los años. Debido a su doble naturaleza se expondrá tanto el estado del arte de las semánticas de procesos, como el de la teoría de categorías, con especial hincapié (sobre todo) en este segundo caso en el mundo de las coálgebras. Terminamos la sección con una muy breve exposición de las aportaciones de la tesis, que los enlaza con los citados trabajos previos, los objetivos concretos que nos propusimos resolver en la tesis y, por último, presentamos la estructura del resto de la tesis.

1.1. Semánticas de procesos

Se puede afirmar de manera general que cualquier programa no hace sino manipular ciertos datos (que vienen dados por una representación interna determinada) buscando computar con ellos determinados resultados. Pronto se empezó a considerar que, si se quería estudiar con suficiente rigor esas *computaciones* de los programas, era indispensable trabajar con abstracciones de dichos datos para así poder razonar sobre la corrección de programas más y más complejos o, simplemente, demostrar cualquier propiedad intrínseca de los mismos.

Los primeros *ordenadores* (o *computadoras*, nombre que representa mejor la función de estas máquinas) se construyeron entre las décadas de los 40 y los 50 del pasado siglo XX, tomando como base teórica el formalismo de las *máquinas de Turing* (llamadas así por su creador Alan Turing [Tur36]). Todas estas primeras máquinas (por ejemplo el *ENIAC* de 1946 o el *Ferranti Mark 1* de 1951), al igual que el formalismo de Turing, tenían en común que la manera en que operaban era secuencial (ver, por ejemplo, la referencia clásica de Dijkstra [Dij68]), es decir, eran capaces de hacer únicamente una acción (una evaluación, asignación, cálculo, etc) en cada instante, de manera que el flujo del programa era lineal. Según se fueron mejorando los componentes de los ordenadores, estos fueron haciéndose más rápidos y ya en los años 60, en el ordenador *Atlas* [KPH61], se podían solapar las operaciones de entrada y salida. Se empezaron a considerar así las primeras ejecuciones *concurrentes*, en las que las máquinas eran capaces de efectuar varias acciones "a la vez".

Cabe mencionar la diferencia subyacente entre *paralelismo* y *multitarea*, si bien, a la postre ambas nociones plantean el mismo problema formal de la concurrencia. Mientras que el paralelismo implica que distintas computadoras (o sencillamente, procesadores) se encargan de cómputos distintos que se realizan simultáneamente, la multitarea se basa en que una única computadora dedica distinto tiempo a distintas acciones que van solapando su ejecución.

Este nuevo paradigma de programación concurrente hacía necesaria una teoría que fuese capaz de permitir razonar sobre estos nuevos programas, mucho más complejos que los secuenciales. Ya a principios de los años 70, existían tres métodos formales principales para razonar sobre la corrección de los programas [Bae05]: la semántica operacional, la semántica denotacional y la semántica axiomática, pero a la hora de intentar modelizar programas concurrentes, ninguna de estas tres semánticas resultó ser lo suficientemente expresiva como para proporcionar, de un modo intuitivo, un formalismo adecuado para razonar sobre sus propiedades.

Fue en 1980 cuando Robin Milner publicó el libro "Calculus of Communicating Systems" (CCS) [Mil80] dando así comienzo a la investigación de las *álgebras de procesos*. Los procesos se definían usando una sintaxis algebraica y Milner definió su semántica de manera operacional, apoyándose para tal fin en el formalismo de Plotkin [Plo81] de los *sistemas de transiciones etiquetados* (en inglés *labeled transition systems* y, abreviadamente, LTS), que como el propio Plotkin define, esencialmente son la representación (en forma de grafo) de los pasos elementales (las *transiciones* en este contexto) que van de una configuración (de un proceso) a otra configuración. Además, en aquel libro Milner también definió una relación de equivalencia entre los procesos, de manera que se tenía un criterio para que dos procesos sintácticamente distintos fuesen equivalentes. A esta noción de equivalencia Milner la denominó *equivalencia observacional* pues el criterio para que dos procesos fuesen observacionalmente equivalentes era que su comportamiento (es decir, "aquello que el usuario puede ver") fuese indistinguible.

Justo después de la publicación del libro de Milner, David Park [Par81] formuló el concepto de *bisimulación* que, en realidad, había aparecido ya "agazapado" en diversos trabajos previos [San07]. Este concepto resultó ser la verdadera noción de *equivalencia observacional* que Milner pretendía definir en CCS pero que, por cuestiones técnicas un tanto sutiles, este no había conseguido capturar perfectamente en su primer intento. Años después, en 1989, Milner incorporó la definición de bisimulación en una nueva versión actualizada de su libro "Calculus of Communicating Systems" [Mil89].

El concepto de bisimulación fue uno de los grandes avances en la semántica de procesos y puede ser definido con cierta facilidad en forma de juego: uno de los jugadores (el atacante) trata de probar que los dos procesos son distintos eligiendo astutamente la acción que ejecutará uno cualquiera de los dos procesos, mientras que el otro jugador (el defensor) tratará de contestar en el otro proceso. Si el atacante no es capaz de derrotar a su oponente imposibilitándole a contestar en cierto momento, mostrando así que ambos procesos tienen un comportamiento distinto, entonces queda probada la equivalencia observacional que dice que ambos procesos son indistinguibles. Como se observa, la filosofía de la bisimulación es que si no somos capaces de demostrar que dos procesos son distintos, es porque deben de ser iguales.

Durante esta década de los 80 aparecerán otras álgebras de procesos, siendo una de

las más destacables la presentada por C.A.R. Hoare en otro texto paradigmático: "Communicating Sequential Processes" o CSP [Hoa85], si bien es de destacar la coincidencia de títulos con el artículo bastante anterior [Hoa78], en el que en un marco más aplicado aparecían ya algunos de los conceptos principales de los que provino su formalización posterior arriba citada. Al igual que Milner, Hoare usó una sintaxis algebraica para los procesos pero, en su caso, la semántica estaba definida de manera denotacional, de forma que a cada proceso se le asigna su semántica o "valor" en un dominio adecuado. A la hora de comparar procesos Hoare no usó la bisimulación, sino la semántica de *trazas*. El formalismo de CSP presenta una mayor cantidad de operadores que CCS, pues Hoare deseaba simplificar la presentación de procesos complejos. Es interesante reseñar el uso del operador de punto fijo para definir el significado de los procesos recursivos.

Los desarrollos de Milner y Hoare fueron ciertamente independientes, si bien es evidente que tras las primeras publicaciones de ambos, fueron plenamente conscientes de las ideas y avances que se producian "al otro lado". Pero al tratarse de sendas formalizaciones de los "mismos" procesos concurrentes, cuyo desarrollo demandaba un cierto marco común que facilitara su aplicación en la práctica, sobre todo durante aquellos primeros años fue inevitable una cierta rivalidad basada en la comparación entre ambas álgebras de procesos (CCS y CSP), tratando de discernir cuál era "la mejor". Sin embargo, pronto se encontró que el curioso carácter complementario de ambas, que parecían haber buscado soluciones intencionalmente contrapuestas a cada una de las cuestiones centrales para la formalización algebraica de la concurrencia, no era sino plenamente enriquecedora. Por el contrario, el estudio combinado de ambas aproximaciones ofreció, y ofrece aún hoy en día, la base inestimable para conocer a fondo los entresijos de este mundo tan apasionante y complejo como es el de la concurrencia.

Posteriormente han ido apareciendo otras semánticas de procesos, como la propuesta por Matthew Hennessy en 1988 [Hen88], en la que de un modo indirecto se presentaba la metodología de *testing* por medio de la cual se observa la semántica de los procesos a partir de cómo reaccionan ante una serie de *tests* (o pruebas). En este marco dos procesos se consideran equivalentes si sus respuestas a todos los *tests* son equivalentes. Esta metodología captura quizás de un modo más claro la visión de los procesos como *cajas negras* (*black box* en inglés), bajo la cual los procesos son meros contenedores del "artilugio" que los hace evolucionar, y del que no deseamos conocer ningún detalle interno (como puede ser su estado): solo se puede observar su comportamiento una vez empotrado en el *test* que lo examina.

1.2. Sistemas de transiciones y variantes

Resulta complicado concretar exactamente cuándo nacieron los *sistemas de transiciones* (abreviadamente LTS) que pese (o quizá sea más correcto decir "gracias") a su sencillez se han convertido en un formalismo de gran utilidad en la disciplina (véase la figura 1.1 que recoge la visión esquemática de una transición). Plotkin recogió la definición en su famoso artículo de 1981 "A structural approach to operational semantics" [Plo81, Plo04] (abreviadamente SOS) pero, como él mismo reconoce, la idea no fue suya, ya que, aparte de estar fuertemente inspirada en los autómatas, el grupo de Viena

de Landin ya había usado este formalismo (Landin [Lan66], Ollengren [Oll76] y Wegner [Weg72]). Incluso antes, en la década de los 60, aparecieron las *estructuras de Kripke*, y a partir de entonces han aparecido otras estructuras fuertemente ligadas a los LTS como pueden ser los *sistemas modales* [LT88] o los *sistemas mixtos* [DGG97, Dam96], y más recientemente los *interface automata* [dAH01].

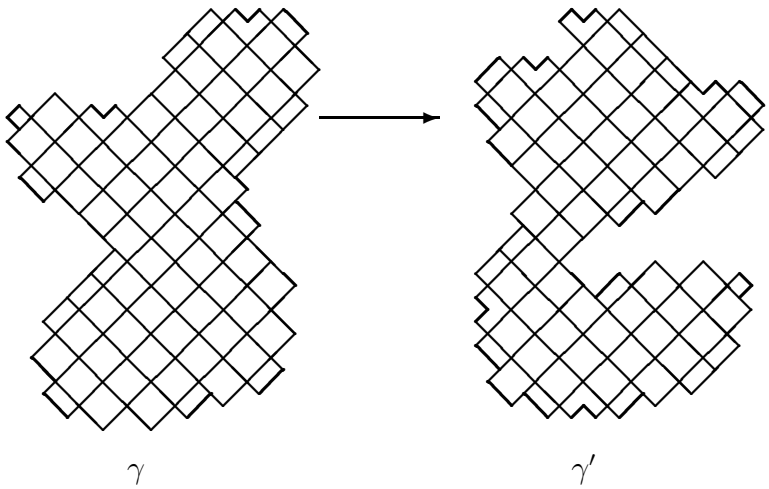


Figura 1.1: La visión esquemática de una transición según Plotkin y publicada en SOS [Plo04].

Los *sistemas de transiciones modales* (abreviadamente MTS y a veces denominados en la literatura *especificaciones modales*) fueron introducidos por Larsen y Thomsen [LT88] en 1988. Larsen y Thomsen consideraban que los LTS no ofrecían suficiente versatilidad a la hora de especificar procesos ya que no permiten distinguir entre comportamientos requeridos y aquellos que son simplemente sugeridos o permitidos. Así nacieron los MTS, donde esencialmente seguimos teniendo un LTS pero con dos tipos distintos de transiciones: las que imponen comportamiento (las denominadas *must*) y aquellas que indican que determinado comportamiento es posible o está permitido (las denominadas *may*). Como resulta natural, en los MTS todo comportamiento impuesto es, en particular, un comportamiento permitido.

Dos años más tarde, en 1990, Larsen y Xinxin [LX90] vieron que los MTS se podían aplicar para resolver algunos tipos de sistemas de ecuaciones. En concreto, los autores demostraron que el conjunto de soluciones de cierto tipo de ecuaciones se podía representar con un *sistema de transiciones modal disyuntivo* en el que se permiten transiciones *must* que modelizan disyunciones entre estados y acciones, las cuales indican que se debe implementar al menos una de ellas. Recientemente, en 2006, Schimdt y Fecher [Sch06a, Sch06b, FS08] presentaron una variante de estos últimos: los denominados *sistemas de transiciones modales one-selecting*. En ellos esa disyunción se interpretaba de manera exclusiva (había que implementar una, y solamente una, de esas disyunciones);

asimismo incluyeron disyunciones con la modalidad *may*.

En 1993, Cerans, Godskesen y Larsen [CGL93] extendieron el modelo de los *sistemas de transiciones modales* añadiéndoles tiempo como una variable real, para definir las *especificaciones modales con tiempo*. Los autores también desarrollaron su propia herramienta de verificación (llamada *EPSION*) para trabajar con este formalismo. En 1996 Dams [DGG97, Dam96] definió los *sistemas de transiciones mixtos* (MiTS) o *especificaciones mixtas*. Al igual que en los MTS, en los MiTS existen dos tipos distintos de transiciones pero, en este caso, las dos no están relacionadas entre sí. De este modo, los *sistemas de transiciones mixtos* generalizan a los *sistemas de transiciones modales*. Pese a que no exista una relación entre los comportamientos permitidos y los impuestos, los *sistemas mixtos*, como se indica en [AHL$^+$08], siguen siendo un modelo útil para modelizar distintas situaciones, como detallaremos en la página 28.

Emparentados en cierto sentido con los sistemas de transiciones modales, están los *interface automata* (o IE de manera abreviada) que Luca de Alfaro y Thomas Henzinger definieron en 2001 [dAH01]. Estos son, a su vez, una sutil variación de los *autómatas con entrada/salida*, que Nancy Lynch definió en [Lyn88], que se diferencian principalmente de los autómatas usuales en que incluyen tres tipos de acciones: entradas, salidas e internas. Esta variación entre los autómatas con entrada/salida y los *interface automata* es esencialmente un cambio de filosofía, que evidentemente acarrea diferencias profundas en la semántica de los dos formalismos. Los autómatas con entrada/salida se centran en una visión "pesimista" del entorno, de manera que se asume que el autómata debe ser capaz de aceptar cualquier entrada en cualquier estado (la condición de *input enabled*), mientras que los *interface automata* asumen una visión "positiva" del entorno, en la que el autómata puede rechazar algunas entradas en determinados estados.

Recientemente, en 2007 Larsen, Nyman y Wasowski [LNW07] unificaron los formalismos de MTS e IE en los denominados *modal I/O automata* que, esencialmente, son *interface automata* en los que sus transiciones tienen las modalidades *may* y *must*.

1.3. Teoría de categorías y coálgebras

La *teoría de categorías* fue introducida en 1945 por Samuel Eilenberg y Saunders MacLane en su artículo "General theory of natural equivalences" [EM45], aunque en realidad algunos de los conceptos básicos como el de *funtor* y *transformación natural* aparecieron en un artículo anterior [EM42] de los mismos autores, con las definiciones limitadas a grupos. De hecho, fue el deseo por parte de Eilenberg y MacLane de abstraer los conceptos en [EM42] lo que les llevo a crear desde prácticamente la nada la teoría de categorías. A la hora de dar nombre a los nuevos conceptos que definieron, Eilenberg y MacLane se inspiraron en la filosofía y, así, tomaron prestada la palabra *funtor* del filósofo Rudolph Carnap [Car], y la palabra *categoría* de los escritos de Aristóteles o Immanuel Kant.

Inicialmente, la teoría de categorías parecía estar más bien abocada a ser un mero lenguaje unificador (una *sintaxis abstracta*, como se denominaba en algunos casos) y, de hecho, durante la primera mitad de los años 50, fue este uso de mero lenguaje matemático el que se le dio a esta nueva teoría, que estaba siendo aplicada durante esos años

principalmente en el campo de la topología algebraica.

Todo cambió cuando en 1957 Alexandre Grothendieck publicó el artículo titulado "Sur quelques points d'algèbre homologique" [Gro57] en el que el papel que desempeñaba la teoría de categorías no era únicamente el de mero lenguaje, sino el de herramienta básica para la construcción de nuevas teorías, que luego el autor aplicó dentro del campo de la geometría algebraica. A Grothendieck le siguió un año después Daniel Kan, que publicó un artículo [Kan58] en el que definió el concepto de *adjunción* para aplicarlo dentro de la teoría de homotopías.

Como decíamos antes, a la hora de razonar sobre un programa debemos trabajar con abstracciones de los mismos. Fue a partir de los años 70 cuando diversos autores vieron que la teoría de categorías podía aplicarse dentro del campo de las ciencias de la computación para obtener diferentes abstracciones de los programas. Como se cuenta en [Jac], Arbib, Manes y Goguen estudiaron el trabajo de Kalman [KFA69] y se percataron de que algunos de sus resultados podían ser generalizados usando teoría de categorías.

Por aquel entonces, Joseph Goguen junto con Thatcher, Wagner y Wright [GTWW77], habían observado que en las ciencias de la computación la proliferación de las distintas semánticas (operacional, denotacional, axiomática, puntos fijos y un largo etcétera), había hecho aparecer demasiados conceptos matemáticos nuevos un tanto extraños, que aunque tuvieran relativa utilidad descriptiva, su base formal última y la relación de las distintas definiciones entre sí no estaban nada claras.

En concreto, podemos citar un párrafo de la introducción del artículo [GTWW77]:

> *In the past few years there has been quite a proliferation of formal semantics for programming languages, or at least of different descriptive terms, for example, operational, interpretive, fixed point, predicate calculus, denotational, algebraic, mathematical, synthesized, W-grammar, axiomatic, inherited, declarative, continuation, process, and now initial algebra semantics. Moreover, mathematical concepts, said to be deep, or strange, or new, are asserted to be relevant, for example, continuos lattices, iterative algebraic theories, infinitary logic, and bicategories. This is quite perplexing. How do these things fit together, if at all? In fact, what is "syntax"; what is "semantics"?*

En la anterior cita, queda de manifiesto que a los autores de [GTWW77] les preocupaba especialmente que, pese a toda aquella proliferación de conceptos, aún no estuviese suficientemente claro qué era la *sintaxis* y qué la *semántica*. Para relacionar la sintaxis y la semántica, los autores trabajaron con *álgebras*, explotando la propiedad de *inicialidad* de estas. Esta propiedad de inicialidad viene a decir que, dentro de la familia de todas las álgebras con unas determinadas propiedades "sensatas", existe siempre un álgebra que es, en cierto sentido, la más pequeña (el *álgebra inicial*), cumpliendo la propiedad de que para cualquier otra álgebra de la familia existe un único morfismo que llega a esta partiendo del álgebra inicial (o sea, que podemos distinguir en toda álgebra mayor una parte que se corresponde exactamente con la inicial).

Para Goguen y el resto de autores de [GTWW77], la sintaxis de una estructura de un lenguaje de programación (por ejemplo un *array* o una *lista*) no era más que un *álgebra inicial*, y cualquier otra álgebra de esa clase definía la semántica de esa estructura. Ese único homomorfismo desde la *sintaxis* de la estructura hasta su *semántica* define la

función semántica. Así nació la *especificación algebraica*, que en particular nos permite referirnos a todos los *tipos abstractos de datos* que se manejan en programación sin necesidad de preocuparnos de ninguna representación interna concreta.

Una década después, en 1980, Arbib y Manes publicaron "Machines in a category" [AM80], en donde aparecen por primera vez los conceptos abstractos de estado, comportamiento o alcanzabilidad, entre otros, pese a que la propia noción de *coálgebra* no apareció aún de forma explícita.

Para terminar, sería inexcusable no referenciar el libro "Non-well-founded Sets" publicado en 1988 y escrito por Peter Aczel [Acz88]. Como el propio Aczel comenta, la inspiración le llegó al leer a Milner y su trabajo sobre CCS. Los procesos de CCS podían tener transiciones circulares, de manera que al intentar otorgarles semántica, esta no se podía definir usando simplemente el concepto de límite, ya que se vulneraba el *axioma de fundación*. El axioma de fundación es uno de los pilares de la teoría de conjuntos clásica y viene a decir que para cada conjunto x no existe una cadena infinitamente descendente de conjuntos

$$\ldots \in x_k \in \ldots \in x_2 \in x_1 \in x_0 = x\,.$$

La propuesta de Aczel fue definir el *axioma de anti-fundación*, es decir, el axioma dual al de fundación (ver [RT92] para una presentación formal de esta dualidad entre los axiomas). A partir de este nuevo axioma, Aczel definió también el concepto dual al de álgebras, dando lugar a las denominadas *coálgebras*[1]. Aparecen de inmediato las *coálgebras finales*, que de nuevo, son el concepto dual al de las álgebras iniciales. Además, si la manera de razonar sobre álgebras es la *inducción* apoyándose en el axioma de fundación, Aczel caracterizó a partir del axioma de anti-fundación el concepto de *coinducción*. Una instancia concreta de las definiciones mediante coinducción resultó ser la definición de bisimulación de Park [Par81]. Estrictamente hablando, la generalización al mundo categórico del concepto de bisimulación apareció un año después, pese a que en espíritu ya estaba en la obra de Aczel, en el corto artículo [AM89]. En dicho trabajo, el propio Aczel, con la ayuda de Mendler, demostró el teorema de existencia de coálgebras finales (enunciado en [Acz88]) y se añadía, además, la citada definición de bisimulación para coálgebras, como un caso de coinducción.

A lo largo de estos años han aparecido muchos otros artículos que aplican la teoría de categorías a las ciencias de la computación. En el contexto de esta tesis, caben destacar los trabajos de Rutten que aportan los fundamentos de las semánticas basados en coálgebras [RT93, TR98] o sus trabajos sobre la coálgebra universal [Rut00]. Además, son especialmente interesantes los tutoriales de Rutten y Bart Jacobs [JR97, Jac] sobre coálgebras y coinducción, así como las referencias que se encuentran en ellos.

1.4. Esta tesis

Inicialmente esta tesis nació a partir de mi interés en conocer más profundamente los formalismos teóricos que podían aplicarse al estudio de la informática. En mi primer año en el programa de doctorado de la Facultad de Informática, David de Frutos me recomendó que leyera una serie de tutoriales que trataban el tema de las coálgebras,

[1] En este caso concreto las F-coálgebras.

que él mismo estaba estudiando en aquellos momentos. Se trataba en concreto de los tutoriales de Rutten [Rut00, Rut01, JR97]. En [Rut01], pese a centrarse más en el uso de las cadenas (*streams*, en inglés), se ponía de manifiesto la naturalidad del uso de las coálgebras para el estudio abstracto de los programas; mientras que en [JR97] y [Rut00] se hacía un estudio más profundo y general de los conceptos básicos de coálgebras/álgebras, finalidad/inicialidad y coinducción/inducción. En aquellos primeros pasos también me resultó especialmente útil el libro que estaba escribiendo Jacobs [Jac] (iniciado en 2005 y que debido a su exceso de celo en abarcar por completo el tema aún hoy sigue en desarrollo). Ahí ya pude ver cómo de un modo relativamente sencillo se podían definir los *sistemas de transiciones*, los *autómatas* y otros muchos formalismos abstractos de la teoría de la computación en el marco coalgebraico.

Además, en el tutorial de Jacobs [Jac], aparecía una definición coalgebraica alternativa de bisimulación y se incluían (aunque muy de pasada) algunos detalles de la definición coalgebraica de simulación. Aquello parecía una vía muy interesante y prometedora de estudio. David de Frutos estaba dirigiendo a la vez la tesis de Carlos Gregorio Rodríguez, una de cuyas líneas de trabajo estaba dedicada a la unificación del espectro *linear time-branching time* de van Glabbeek [vG01], por lo que además pareció que el estudio coalgebraico podría complementar ese trabajo.

Mi producción científica comenzó con mi trabajo de tercer ciclo [Fáb06], donde tratamos de trasladar los resultados clásicos de reflexión y preservación de propiedades lógicas vía bisimulaciones [vB76] y el de reflexión vía simulaciones usando el marco categórico. Buena parte de la inspiración para obtener los resultados de reflexión en simulaciones vino de la tesis de mi co-director, Miguel Palomino [Pal05]. Si bien los resultados clásicos se pudieron trasladar satisfactoriamente al marco coalgebraico en el caso de las bisimulaciones, tuve que afrontar mi primer (serio) obstáculo a la hora de estudiar en qué circunstancias las simulaciones categóricas reflejaban (o preservaban) las propiedades lógicas.

Mis directores y yo llegamos a la conclusión de que la excesiva generalidad de la definición de simulación categórica propuesta por Hughes y Jacobs [HJ04] era (en cierta medida) la "culpable" de que solo hubiésemos podido obtener resultados parciales. Surgió así la búsqueda de una noción más restrictiva, aunque todavía suficientemente general de simulación categórica, que facilitara a las mismas unas propiedades comunes mejores.

Junto a esta búsqueda continuamos con la tarea de unificación de las semánticas lineales, ahora desde el enfoque categórico. La idea fue, justamente, aprovechar la amplitud de la definición de simulación coalgebraica para cubrir diversas nociones de simulación que capturasen nuevas propiedades interesantes dentro del marco de los sistemas de transiciones. Al capturar dichas nociones dentro del marco general se simplificaron o incluso se hicieron innecesarias las demostraciones de los resultados sobre las mismas, al poder ser abordadas de una vez por todas al estudiar los resultados en el marco general.

De esta manera quedaron construidos los hilos conductores de la tesis en la que nos propusimos un viaje a dos bandas: una categórica y general; y otra algo más concreta y aplicada, dedicada al estudio de nuevas semánticas de procesos.

1.4.1. Objetivos de la tesis

De una manera global y general se puede decir que el terreno en el que se mueve la presente tesis es el del estudio general de las semánticas de procesos en el marco categórico, haciendo un especial hincapié en el estudio de las relaciones entre estas; es decir, un estudio de las relaciones de bisimulación y simulación (o abreviadamente, como muchas veces escribimos, (bi)simulación).

En la sección 2.1 veremos los principales conceptos del mundo clásico de las semánticas de procesos, y resumiremos brevemente las contribuciones principales de autores como Hoare, Milner o Hennessy, para posteriormente, ver con un mayor detalle los distintos modos de describir el comportamiento de los procesos. Si bien no pretendemos tratar de un modo exhaustivo todas las presentaciones posibles, sí que explicaremos algunas de las más utilizadas, como son los sistemas de transiciones (sección 2.1.1) y los sistemas modales (sección 2.1.4), y trataremos también diversas estructuras que manejan acciones de entrada y salida (sección 2.1.3) o las estructuras de Kripke y los sistemas de transiciones probabilísticos (sección 2.1.2). A lo largo de las descripciones de estos sistemas destacaremos las dos maneras canónicas que existen para comparar procesos: la equivalencia de bisimulación y el orden de simulación. Pero, como veremos, a lo largo de la literatura estas definiciones se han realizado de manera *ad-hoc* para cada tipo de estructura (LTS, estructuras de Kripke, etc), sin disponerse de un marco común que facilitara el estudio de todas ellas a un mismo tiempo.

Por otro lado, dedicaremos la sección 2.2 a la presentación de los conceptos básicos de la teoría de categorías que hemos aplicado a lo largo de nuestra investigación y, más concretamente, a los que se refieren a las coálgebras. Tampoco haremos un recorrido exhaustivo de la disciplina, pues basta echarle un vistazo a libros como "Category theory for the working mathematician" [Mac98] para comprobar lo amplio de este mundo categórico y la ingente cantidad de conceptos que en él se pueden formalizar, por lo que nos centraremos únicamente en aquellos que se utilizan en nuestras publicaciones. Dentro de este marco categórico veremos los distintos modos existentes de generalizar la noción de bisimulación (sección 2.2.2) y simulación (sección 2.2.3) y cómo, en particular, el marco categórico permite tener un único concepto (general) de (bi)simulación, cuyas instancias (particulares) definen las nociones concretas para LTS, estructuras de Kripke, sistemas probabilísticos, etc.

Además, prestaremos un especial cuidado al explicar cómo la semántica de procesos clásica de CCS motivó a Peter Aczel para definir las F-coálgebras y la bisimulación entre LTS, como una instancia del principio de coinducción (página 42). De este modo quedará patente lo estrecho que es el nexo que une al mundo de las semánticas de procesos con la teoría de categorías. Por lo tanto, resulta lógico estudiar más en profundidad este nexo de unión entre estas dos teorías, que comparten tantos puntos vecinos (bisimulación/coinducción, sistemas de transiciones/coálgebras, etc). Precisamente fue el deseo de comprender mejor esta relación el que nos propusimos saciar al abordar la presente tesis.

Para entender mejor la relación entre las coálgebras y las semánticas de procesos, decidimos empezar estudiando las propiedades esenciales de las relaciones de (bi)simulación entre coálgebras. El interés de los posibles resultados que pudiésemos obtener con nuestro estudio entendíamos que radicaría en el alcance categórico de la noción de coálgebra,

que hace que no estemos trabajando exclusivamente con sistemas de transiciones, sino con toda la familia de estructuras matemáticas que pueden definirse como coálgebras (sistemas de Kripke, autómatas, árboles, etc). Por ello nos propusimos mantener siempre la mayor generalidad posible, para así abarcar la mayor colección posible de situaciones.

La piedra angular de nuestra investigación la han constituido las (bi)simulaciones categóricas. Aunque existen distintas aproximaciones a su definición, al final todas resultan ser equivalentes (como en el caso de la bisimulación) o coinciden en los casos más interesantes, que además son los que tienen buenas propiedades (el caso de la simulación). Por otro lado mientras que el concepto de bisimulación coalgebraica es, en cierto sentido, preciso y único siendo una clara generalización del correspondiente para los sistemas de transiciones, el de simulaciónes es sin duda alguna mucho más "abierto", pues si aceptamos la noción de simulación categórica de Hughes y Jacobs, no se puede garantizar siquiera que las nociones de similitud inducidas sean siempre transitivas.

Un rápido vistazo a la noción categórica de simulación nos permitió ya observar que si bien el concepto clásico de simulación puede verse como "la mitad" de la bisimulación, en el mundo categórico la simulación incluye como caso particular en su definición a la propia bisimulación. Era esta generalidad de la definición de simulación de Hughes y Jacobs (aunque quizás llegue a ser excesiva) la que, desde nuestro punto de vista, hacía especialmente interesante su estudio, por lo que todos nuestros esfuerzos se concentraron en ella durante la primera parte de la investigación.

En concreto, parecía interesante estudiar en qué casos las instancias de dichas nociones categóricas generales de (bi)simulación permitían traspasar propiedades entre las estructuras relacionadas. Contando con los resultados previos de mi co-director Miguel Palomino en su tesis [Pal05], resultó natural centrarnos en las propiedades que pueden expresarse como fórmulas lógicas. Surgió así el primer objetivo de la tesis:

1 *Estudiar si la simulación coalgebraica se comporta del mismo modo que la clásica en relación a sus propiedades lógicas; y en qué medida podemos aislar nociones de simulación dentro de este contexto categórico que compartan todas las propiedades deseables de la semántica clásica.*

La segunda gran vía de nuestra investigación surgió de manera natural de la primera. Una vez que nos habíamos propuesto el estudio detallado de la simulación categórica de Hughes y Jacobs, comprobamos que esta admitía como casos particulares una gran variedad de semánticas, además de la generalidad de las estructuras sobre las que estas se pueden aplicar.

Partiendo de esta constatación elaboramos nuestro segundo objetivo que, a su vez, puede dividirse en dos puntos:

2.1 *La presentación unificada de las distintas nociones de simulación previamente conocidas dentro del marco categórico.*

2.2 *El estudio de nuevas semánticas de simulación entre procesos.*

Los distintos tipos de sistemas de transiciones, junto con las relaciones de (bi)simulación que veremos en la sección 2.1, son la base de este estudio, que nació como consecuencia de nuestro deseo original de encontrar una noción canónica de simulación (categórica),

pero luego se desligó en parte de este marco general para centrarse en el estudio de ciertas semánticas de procesos.

Organización de la tesis

De acuerdo con el punto 4,4 de la Normativa reguladora de estudios universitarios oficiales de postgrado de la Universidad Complutense, la presente tesis se acoge al formato de colección de publicaciones. Existen dos motivos principales para que entendamos que esta es la decisión adecuada. En primer lugar prácticamente toda la investigación que hemos realizado en la materia ha sido ya publicada en las actas de congresos internacionales de reconocido prestigio, lo que garantiza que el trabajo aquí presentado ha sido no solo revisado por mis directores y yo mismo, sino que también ha pasado por las revisiones de los congresos y, además, ha sido ampliamente discutido en los mismos. En segundo lugar, la propia colección de publicaciones tiene suficiente cohesión y relación entre sí como para justificar que no estamos ante una mera colección de publicaciones dispares, sino que podría verse perfectamente como la colección (casi) completa de publicaciones a las que podría haber dado lugar una tesis presentada conforme al formato "clásico".

Previamente a la presentación de esos artículos, en el capítulo 2 veremos una introducción de los conceptos básicos necesarios para la correcta comprensión de los mismos. Haremos especial hincapié en los dos pilares esenciales de la tesis: las semánticas de procesos (sección 2.1) y la teoría de las coálgebras (sección 2.2). En cada uno de ellos discutiremos todos aquellos aspectos que se relacionan con las publicaciones que conforman el núcleo de la tesis.

En el capítulo 3, detallamos los resultados y la relación existente entre las distintas publicaciones que componen la tesis. El siguiente capítulo 4 propone una pequeña mirada a lo que esperamos sea la continuación de nuestra investigación en el futuro cercano, presentándose las conclusiones de la tesis.

Por fin, en el capítulo 5, presentamos las siete publicaciones que conforman el núcleo de la tesis. Estas se disponen en dos bloques para señalar su diferente carácter de acuerdo a los dos objetivos centrales de la tesis: las tres primeras son las publicaciones de corte más categórico, mientras que las cuatro restantes son las publicaciones que hablan sobre las nuevas semánticas de simulación que hemos definido.

Por comodidad a lo largo de la tesis (y más concretamente en el capítulo 3) nos referiremos a los artículos con los siguientes nombres:

Publicaciones del bloque categórico:

- C1: "Reflection and preservation of properties in coalgebraic (bi)simulations". Publicado en el congreso ICTAC en 2007. Página 87,
- C2: "Non-Strongly Stable Orders Also Define Interesting Simulation Relations". Publicado en el congreso CALCO en 2009. Página 102, y
- C3: "Multiset bisimulations as a common framework for ordinary and probabilistic bisimulations". Publicado en el congreso FORTE en 2008. Página 117.

Publicaciones del bloque que trata sobre nuestras nuevas semánticas de simulación:

- S1: "Relating modal refinements, covariant-contravariant simulations and partial bisimulations". Publicado en el congreso FSEN en 2011. Página 133,
- S2: "Graphical representation of covariant-contravariant modal formulas". Publicado en el congreso EXPRESS en 2011. Página 149,
- S3: "Logics for Contravariant Simulations". Publicado en el congreso FMOODS-FORTE en 2010. Página 164, y
- S4: "Equational Characterization of Covariant-Contravariant Simulation and Conformance Simulation Semantics". Publicado en el congreso SOS en 2010. Página 172.

Finalmente, en el Apéndice A se incluyen las versiones extendidas de algunas de nuestras publicaciones, en las cuales se expanden algunas demostraciones y se detallan algunos de los resultados de las versiones publicadas. En todos los casos estas versiones extendidas fueron en realidad nuestros trabajos de base, de las que en función de las limitaciones habituales de espacio en las actas de los congresos donde se publicaron, se extrajeron las versiones publicadas. Por ello mismo sirvieron de hecho como ratificación de los resultados presentados en los artículos, siendo facilitadas en su día a los revisores junto con las versiones publicadas para que pudieran examinarlas si lo consideraban oportuno a la hora de valorar los mismos.

Capítulo 2

Conceptos fundamentales previos

En el presente capítulo expondremos las definiciones y conceptos básicos necesarios para la correcta comprensión de la investigación que ha dado pie a esta tesis.

Actualmente, la informática se ha convertido en una importantísima herramienta en el día a día de cualquier persona y los ordenadores ya no se utilizan únicamente en el campo militar (para tratar de descifrar códigos enemigos, como en sus orígenes). Desde los actuales móviles, con potencia superior a los ordenadores convencionales de hace cinco años, hasta los PC's, estamos tan rodeados de computadoras que ya no es raro encontrarse con noticias en las que diversas desgracias se achacan a los denominados *errores informáticos* (caídas de sistemas, comportamientos erróneos no esperados, es decir, los denominados *bugs*, etc) que hacen perder millones de euros a las empresas de todo el mundo. Por lo tanto, cada vez tiene más sentido el estudio formal de la computación que, como meta más general, trata de ayudarnos a comprender mejor qué y cómo computan los programas informáticos.

De un modo general, esta tesis puede englobarse dentro del estudio de los sistemas de procesos y, muy especialmente, del estudio de las relaciones entre esos procesos. Este estudio está además sustentado en gran parte por la generalidad y potencia de la *teoría de categorías* y, más concretamente, de las *coálgebras*. Por ello, este capítulo introductorio comienza explicando los conceptos básicos de la teoría de procesos, así como distintas maneras de representarlos, como pueden ser los *sistemas de transiciones*, para posteriormente introducir todos aquellos conceptos de la *teoría de categorías* y las *coálgebras* que nos permitirán extender muchos de los conceptos anteriores a este marco más general y elegante.

2.1. Semánticas de procesos

Como comentábamos en el capítulo 1, la teoría de las semánticas de procesos surgió al final de la década de los 60 del pasado siglo, ante la necesidad de proporcionar una teoría que permitiese un estudio satisfactorio de los procesos concurrentes. Aquellos primeros trabajos corresponden a los autores Edsger W. Dijkstra [Dij68], en cuanto al estudio de los procesos secuenciales, pero ya con ideas del nuevo formalismo de los procesos concu-

rrentes, y C.A.R. Hoare [Hoa78], que ya definió y trabajó con este nuevo tipo de procesos. En estos primeros artículos los procesos se ven como la mera descripción de un comportamiento que, a su vez, puede dividirse en otros subprocesos (o subcomportamientos) eventualmente ejecutables en paralelo.

Evidentemente, según fueron pasando los años el concepto de proceso fue puliéndose y el modelo se fue ampliando para incluir operadores que permitiesen trabajar con el tiempo, probabilidades, prioridades, etc. Nosotros no pretendemos hacer un repaso exhaustivo de las semánticas de procesos propuestas a lo largo de la literatura, sino centrarnos en la manera en que la semántica operacional de un proceso puede ser definida. Por ello, en esta sección vamos a resumir todos aquellos conceptos referentes a los sistemas de transiciones, así como algunas de sus variantes, en especial los conceptos referentes a los sistemas de transiciones modales y mixtos.

2.1.1. Sistemas de transiciones y (bi)simulaciones

Gordon Plotkin publicó en 1981 el artículo "A structural approach to operational semantics" [Plo81, Plo04] en el que se introducen por primera vez los sistemas de transiciones como representación de los procesos, proporcionando la semántica operacional (formal) de los mismos. Aunque ya hemos comentado en la sección 1.2 que esta manera de representar procesos no era nueva, lo que sí que fue novedoso fue que Plotkin definió las transiciones de los sistemas por medio de una serie de reglas estructuradas, de manera que el comportamiento del programa completo estaba caracterizado por el comportamiento de sus partes (o sea, de un modo composicional). Este formalismo disponía de una serie de operaciones definidas en el plano sintáctico que permitía la composición para dar lugar a otras operaciones. Podemos decir que la semántica operacional de un programa define de un modo formal la secuencia de pasos (o transiciones) que se han de dar para su ejecución.

Aunque normalmente a lo largo de la tesis hablaremos sencillamente de *sistemas de transiciones*, en realidad con este nombre nos referiremos a la noción clásica de los *sistemas de transiciones etiquetados* (o, abreviadamente, LTS del inglés *Labeled Transition Systems*), en los que cada transición tiene uno o varios nombres (o *etiquetas*) que corresponden a cada acción a perteneciente a un cierto alfabeto A. Es decir:

Definición 2.1.1. *Un* sistema de transiciones etiquetado*, o* LTS*, es una terna* $\mathcal{A} = (X, A, \longrightarrow)$ *donde* X *es su conjunto de estados,* A *es su conjunto de etiquetas o acciones, y* $\longrightarrow \subseteq A \times X \times X$ *es su relación de transición.*

Generalmente se denota por $x \xrightarrow{a} x'$ *el que* $(a, x, x') \in \longrightarrow$*, en cuyo caso diremos que* x' *es un* a*-sucesor de* x*.*

Destacamos que los LTS finitos sin bucles y finitamente ramificados, es decir, aquellos en los que X es finito, ninguno de sus estados tiene una cantidad infinita de sucesores y desde ninguno de los cuales puede regresarse al mismo estado vía una secuencia de transiciones, pueden describirse sintácticamente por medio del álgebra de procesos BCCSP cuya definición damos a continuación.

Definición 2.1.2. *Dado un conjunto de acciones Act, el conjunto de procesos p en forma* BCCSP *se define con la gramática expresada en forma de Bakus-Naur por medio de*

$$p ::= 0 \mid ap \mid p + p\,,$$

donde $a \in Act$.

0 *representa al proceso que no ejecuta ninguna acción (en otras álgebras representado por* stop *o* nil*), ap es el prefijo secuencial de acciones y* $+$ *es el operador de elección.*

La semántica de los procesos BCCSP se define por medio de las siguientes reglas:

$$ap \xrightarrow{a} p \qquad \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

El tratamiento de los procesos infinitos e infinitamente ramificados es, como parece lógico, bastante más complicado. Ya Hoare en CSP [Hoa85] introdujo el operador de punto fijo *fix* para definir procesos infinitos. Podemos considerar que los procesos así definidos se corresponden con "la menor solución" de determinada ecuación. Más adelante, en la sección 2.2, veremos cómo las coálgebras nos permiten trabajar de un modo sencillo con procesos infinitos.

Bisimulaciones

Cuando Plotkin introdujo las semánticas operacionales en 1981 [Plo81], una de sus preocupaciones fue la de que según se observase más o menos un sistema de transiciones, la interpretación que obteníamos del mismo podía cambiar sustancialmente. Del mismo modo Robin Milner consideró ya en 1980 [Mil80] en su primera versión de CCS que dos procesos podían ser sintácticamente muy distintos pero sus comportamientos idénticos. Su noción de *comportamientos idénticos* quedaba formalizada en dicho texto mediante la siguiente definición inductiva de *equivalencia observacional*.

Definición 2.1.3. *Sea* $\mathcal{A} = (W, A, \longrightarrow)$ *un LTS. Se define la* semántica observacional *sobre su conjunto de estados* W $\sim_\omega$ *de la siguiente manera:*

- $\sim_0 = W \times W$.
- $s \sim_{n+1} t$ *para* $n \geq 0$ *si:*
  1. *para todo* s' *con* $s \xrightarrow{a} s'$, *existe un* t' *tal que* $t \xrightarrow{a} t'$ *y* $s' \sim_n t'$, *y*
  2. *para todo* t' *con* $t \xrightarrow{a} t'$, *existe un* s' *tal que* $s \xrightarrow{a} s'$ *y* $s' \sim_n t'$.
- $\sim_\omega = \bigcap_{n \geq 0} \sim_n$.

Nota 2.1.1. *La idea intuitiva que Milner pretendía capturar era que* $p \sim_\omega q$ *se tendría si fuera imposible distinguir los procesos p y q en un número finito de pasos.*

Robin Milner terminaba aquella primera versión de CCS en la Universidad de Edimburgo, en la que David Park iniciaba un año sabático. Park era un experto en las teorías de punto fijo y, gracias a ello, se percató de que aquella primera definición de equivalencia observacional que hizo Milner no resultaba adecuada en todos los casos, al no poder ser

presentada mediante un punto fijo. Para arreglar este pequeño (pero siempre importante) error, Park propuso la siguiente formulación coinductiva de la semántica observacional que Milner incluyó en su posterior edición de CCS en [Mil89].

Definición 2.1.4. *Dados dos sistemas de transiciones etiquetados* $\mathcal{A} = (X, A, \longrightarrow)$ *y* $\mathcal{B} = (Y, A, \longrightarrow)$*, una bisimulación* R *es una relación binaria* $R \subseteq X \times Y$ *tal que si* xRy*:*

- *Si* $x \xrightarrow{a} x'$ *entonces existe* $y' \in Y$ *tal que* $y \xrightarrow{a} y'$ *con* $x'Ry'$*.*
- *Si* $y \xrightarrow{a} y'$ *entonces existe* $x' \in X$ *tal que* $x \xrightarrow{a} x'$ *con* $x'Ry'$*.*

A la unión de todas las bisimulaciones se le llama bisimilaridad.

La definición inicial de *equivalencia observacional* propuesta por Milner no coincide con la bisimulación, aunque no por el hecho puramente cosmético de que en su definición se manejen dos LTSs en lugar de uno solo. Veamos a continuación un ejemplo clásico extraído de [San07] con el que queda de manifiesto además que la noción adecuada de equivalencia observacional es la bisimulación, y no la equivalencia de la definición 2.1.4.

Ejemplo 2.1.1 Siendo $a \in Act$, definimos los siguientes estados:

- x^0 es un estado sin transiciones.
- Para $n \geq 1$, x^n es un estado que tiene como única transición $x^n \xrightarrow{a} x^{n-1}$.
- x^ω es el estado que tiene como única transición un ciclo: $x^\omega \xrightarrow{a} x^\omega$.
- s es un estado que tiene como transiciones $s \xrightarrow{a} x^n$, para cada $n \geq 0$.
- t tiene como transiciones $t \xrightarrow{a} x^n$, para cada $n \geq 0$; y además, $t \xrightarrow{a} x^\omega$.

Por inducción se puede probar que para todo n se tiene que $s \sim_n t$ y, por lo tanto, también tenemos que $s \sim_\omega t$. Ahora bien, es evidente que s y t no son bisimilares ya que, para todo n, el estado x^ω puede realizar una cantidad arbitraria de a-transiciones, mientras que cada x^n solo puede hacer $n + 1$ a-transiciones.

En el ejemplo 2.1.1 se ha tenido que recurrir a dos LTS que no son finitamente ramificados; en general, si en la definición 2.1.3 de equivalencia observacional se reemplaza la ω-inducción por una *inducción transfinita* entonces sí que es cierto que en el límite se obtiene la bisimulación. De hecho, en el caso de los LTS finitamente ramificados, las definiciones 2.1.4 y 2.1.3 coinciden.

Como también se comenta en [San07], hay dos detalles muy interesantes a notar en la definición de bisimulación: en primer lugar, es una definición puramente local y en segundo lugar, no hay ningún tipo de jerarquía entre los pares de la bisimulación. Decimos que la definición es local puesto que para cada estado únicamente debemos comprobar sus sucesores inmediatos, sin necesidad de preocuparnos por el resto. Esto es diametralmente opuesto, por ejemplo, a la equivalencia de trazas que viene dada por una definición global, puesto que dos procesos son equivalentes en trazas si ambos tienen los mismos cómputos, lo que significa que a partir de cada estado podemos tener que

mirar los sucesores de otros muchos. La no existencia de una jerarquía entre los pares de la bisimulación se traduce en que a la hora de comprobar si dos estados son bisimilares basta con construir una bisimulación que contenga a dicho par de estados.

Como veremos en la sección 2.1.2 (página 19), la bisimulación también apareció en los años 70 dentro del marco de las lógicas modales. Quince años después, Matthew Hennessy y Robin Milner presentaron en [HM85] una lógica que caracterizaba a la bisimulación, es decir, una lógica tal que dos procesos son bisimilares si, y solo si, ambos cumplen las mismas fórmulas. La lógica de Hennessy-Milner sobre LTS quedó definida por medio de las siguientes reglas:

Definición 2.1.5 ([HM85])**.** *Dado un LTS sobre un alfabeto I,* la lógica de Hennessy-Milner $\mathcal{L}_{HM}$ *de fórmulas sobre I está definida recursivamente como:*

- tt *está en* $\mathcal{L}_{HM}$*.*
- *Si* $A, B \in \mathcal{L}_{HM}$*, entonces* $A \wedge B, \neg A \in \mathcal{L}_{HM}$*.*
- *Si* $A \in \mathcal{L}_{HM}$*,* $i \in I$*, entonces* $\langle i \rangle A \in \mathcal{L}_{HM}$*.*

La relación de satisfacción $\models$ *es la menor relación tal que:*

- $p \models \mathsf{tt}$*, para todo* p*.*
- $p \models A \wedge B$ *si* $p \models A$ *y* $p \models B$*.*
- $p \models \langle i \rangle A$ *si existe algún,* i*-experimento* p' *(es decir* $p \xrightarrow{i} p'$*) con* $p' \models A$*.*

Por comodidad, los autores consideraban también las siguientes operaciones derivadas:

Definición 2.1.6 ([HM85])**.** *Se definen los siguientes operadores derivados en la lógica de Hennessy-Milner:*

- ff *se define como* $\neg\mathsf{tt}$*.*
- $A \vee B$ *se define como* $\neg(\neg A \wedge \neg B)$*.*
- $\langle s \rangle A$ *se define como* $\langle i_1 \rangle \dots \langle i_n \rangle A$*, donde* $s = i_1 \dots i_n$ *con* $n \geq 1$*.*
- $[s]A$ *se define como* $\neg\langle s \rangle \neg A$*.*

Finalmente, para los procesos BCCSP de la definición 2.1.2 existe una caracterización axiomática de la bisimulación. En concreto, se tiene que dos procesos p y q son bisimilares si, y solo si, se puede derivar su igualdad a partir del conjunto de axiomas $\{\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3, \mathsf{B}_4\}$ definidos en la figura 2.1.

Simulaciones

Milner también definió el concepto de simulación que puede ser visto como "la mitad" de la bisimulación, en la que únicamente se exige que el proceso que simula sea capaz de hacer todo lo que hace el otro proceso. Formalmente, tenemos la siguiente definición:

(B_1) $x + y = y + x$.

(B_2) $(x + y) + z = x + (y + z)$.

(B_3) $x + x = x$.

(B_4) $x + 0 = x$.

Figura 2.1: Axiomatización de la bisimulación para procesos BCCSP.

Definición 2.1.7. *Dados dos sistemas de transiciones etiquetados* $\mathcal{A} = (X, A, \longrightarrow)$ *y* $\mathcal{B} = (Y, A, \longrightarrow)$*, una* simulación *entre ellos es una relación binaria* $H \subseteq X \times Y$ *tal que si* xHy*:*

- *Para cada* $x \xrightarrow{a} x'$*, existe un* $y' \in Y$ *tal que* $y \xrightarrow{a} y'$*, con* $x'Hy'$*.*

A la unión de todas las simulaciones se le llama relación de similaridad o de similitud.

Pero a pesar de los parecidos entre la formulación de la simulación y de la bisimulación, la similitud es una relación mucho más alejada de la bisimilaridad de lo que a priori pudiéramos pensar. Por supuesto, para empezar la similitud es solo un orden, mientras que la bisimilaridad es una relación de equivalencia. Además, aunque podría parecer en principio que el núcleo de la similaridad (es decir, la relación resultante de intersecar la similaridad y su opuesta) generaría la bisimilaridad esto no es en absoluto así. De hecho este núcleo o equivalencia de simulación es una equivalencia sí, pero que distingue mucho menos pares de procesos que la bisimilaridad. Es más: no existe ningún orden (no trivial) cuyo núcleo genere la bisimulación.

La clave por la que se tiene lo anterior, es que existe una simulación mutua entre dos procesos p y q siempre que p simule a q y q simule a p; pero no se exige que la simulación entre p y q sea la inversa de la simulación entre q y p (véase, por ejemplo [vG01], para encontrar un sencillo contraejemplo).

La simulación también se puede caracterizar lógicamente (ver de nuevo [vG01], entre otros). En general se dice que una lógica $\mathcal{L}$ caracteriza una relación de orden $\lesssim$, y por tanto la semántica inducida por ella, si para todo par de procesos se tiene $p \lesssim q$ si, y solo si, $p \models \varphi \Rightarrow q \models \varphi$, para toda fórmula $\varphi \in \mathcal{L}$; en palabras llanas, q será mayor o igual que p si q cumple todas las fórmulas que cumple p. Para el caso de la simulación estándar una lógica que la caracteriza es $\mathcal{L}_S = \{\mathsf{tt}, \bigwedge_{i\in I} \varphi_i, \langle a\rangle\varphi\}$.

Para finalizar este apartado, destacamos la reciente investigación de Baeten y otros [BvBL$^+$10] que les ha llevado a definir una noción de simulación que, como explicaremos en la sección 3.3, guarda una relación muy cercana con nuestra investigación. Esta noción es la *bisimulación parcial* sobre LTS, y que recibe su nombre de combinar la condición de bisimulación y la de simulación. En concreto, el conjunto de acciones A se parte en B y $A \setminus B$, imponiéndose la condición de bisimulación para las acciones de B y la condición de simulación para el resto. Intuitivamente, las acciones de B son aquellas que corresponden a ciertos eventos que ofrece el entorno y que no pueden ser controlados por el usuario, mientras que aquellas de $A \setminus B$ corresponden con las acciones controlables de la máquina.

2.1.2. Variantes de los sistemas de transiciones

Además de la sencillez de su definición, los sistemas de transiciones son una estructura tremendamente versátil que a lo largo de la literatura especializada ha visto cómo se ha ido modificando para añadir distintas capacidades. Aquí recapitulamos algunas de ellas que hemos utilizando directamente en nuestra investigación, o que están en cierto modo relacionadas con ella, aunque, por supuesto, hay otras muchas variantes (véase, por ejemplo, [AFV01]).

Los sistemas de transiciones ordinarios no incluyen ningún tipo de información adicional sobre las peculiaridades de cada uno de los estados que los forman. En muchas ocasiones, y muy especialmente cuando se quiere estudiar las propiedades lógicas de un sistema, es deseable poder contar con un modelo abstracto que incluya cierta información lógica extra en cada uno de los estados. Así surge una primera variante de los sistemas de transiciones conocida como *estructuras de Kripke*.

En los años 60 del siglo pasado, Saul Kripke fue el primero que decidió añadir a cada estado de un LTS el conjunto de propiedades atómicas que se satisfacían en este. Dado un conjunto AP de proposiciones atómicas, una *estructura de Kripke* es una terna $\mathcal{A} = (X, \longrightarrow, \epsilon)$ en la que $(X, \longrightarrow)$ es un sistema de transiciones (no etiquetado) y $\epsilon : X \to \mathcal{P}(AP)$ es la llamada función de etiquetado o de observación, que asocia a cada estado el conjunto de propiedades atómicas que satisface.

En la literatura clásica sobre el tema (por ejemplo en [CGP99]), a las estructuras de Kripke se les suele exigir que la relación $\longrightarrow$ sea total, es decir, que para cada $x \in X$ exista algún $y \in X$ tal que $x \longrightarrow y$. No obstante, dentro del marco coalgebraico en el que nos hemos movido esta exigencia no es requerida siempre, por lo que nosotros tampoco lo hemos exigido.

La simulación entre estructuras de Kripke está definida como las simulaciones para sistemas de transiciones, donde además se exige que para cada par de estados (x, y) relacionados se tenga que la función de observación $\epsilon(x)$ contenga a $\epsilon'(y)$, es decir, que el estado y no cumpla más proposiciones que el estado x[1]. Análogamente, la bisimulación exige además la igualdad entre las funciones de observación para los pares de estados relacionados.

Como adelantábamos en la sección anterior, en realidad, este concepto de bisimulación también apareció de manera independiente dentro del marco de las lógicas modales ya en los años 70 en la tesis de Johan van Benthem [vB76]. Durante los años 60 una vía de investigación fue el estudio de propiedades *invariantes* entre modelos (que en este contexto eran estructuras de Kripke). Una propiedad entre modelos M y N es invariante si cuando la propiedad es cierta en M entonces también lo es en N (a esta condición se le llama también *preservación* de una propiedad) y viceversa, es decir, si la propiedad es cierta en N también lo es en M (a esta condición se le llama también *reflexión* de una propiedad).

Inicialmente se estudió la manera en que un homomorfismo reflejaba y preservaba propiedades lógicas entre sistemas. Un homomorfismo no es más que una función f entre

[1] Se puede considerar también la igualdad de las funciones de etiquetado en vez de la inclusión. No obstante, como se comenta en [Pal05], la condición más débil que garantiza la reflexión de propiedades es esa inclusión.

un modelo M y otro N, tal que si una proposición se cumple en un estado m de M entonces la proposición $f(m)$ se cumple en N, y si existe una transición entre m y m' en M también existe una transición entre $f(m)$ y $f(m')$ en N.

Como se observa, un homomorfismo tiene bastante más parecido con el concepto de simulación que con el de bisimulación, puesto que no existe la condición recíproca, y esto se traducía en que los homomorfismos no preservaban propiedades y, por tanto, no mantenían los invariantes. Una solución que parecía evidente era la de incluir la condición recíproca, es decir, que si en N hay una transición entre $f(m)$ y cierto n, entonces en M debe existir un m' tal que $m' = f(n)$ y una transición desde m a m'. Esta fue la solución que tomó Kristen Segerberg [Seg68, Seg71] y de esa manera definió los *p-morfismos*. Los p-morfismos son tales que una fórmula φ es cierta en un estado m de M si, y solo si, la fórmula también es cierta en $f(m)$ de N. Eso sí, aún puede suceder que existan dos estados que cumplan las mismas fórmulas pero no exista un p-morfismo entre ellos.

Para solucionar este problema, Benthem definió en su tesis [vB76] las *p-relaciones*, cuya definición coincide con la que dábamos de bisimulación entre estructuras de Kripke. De esta manera se obtiene el resultado clásico que dice que dos estados son bisimilares si, y solo si, cumplen las mismas fórmulas lógicas.

Si avanzamos por la historia, nos encontramos con que en 1997 Alur, Henzinger y Kupferman [AHK97a, AHK97b] definieron una generalización de los sistemas de transiciones a la que llamaron *sistemas de transiciones alternantes* (o ATS del inglés *alternating transition system*). Estas estructuras se conciben como sistemas multi-agentes [Sha53, HF89] en los que la responsabilidad de fijar las transiciones a realizar se distribuye entre dichos agentes (o "partes" del sistema). Una transición en un ATS se realiza de la siguiente manera: dado un estado q, cada uno de los agentes (esencialmente estructuras de Kripke que componen al ATS) elige un conjunto de estados como posibles sucesores. Estos han de verificar, por definición, que sea cual sea la elección de cada agente, la intersección de los conjuntos elegidos será siempre un único estado q'_i, que sería el estado que se tomaría como sucesor de q en la transición "consensuada" entre agentes.

En realidad, la definición general de tales sistemas alternantes no permite encontrar la razón por la que se les llamó alternantes hasta que no se añade la información adicional consistente en un subconjunto A del conjunto de agentes Ω. Entonces, se pueden ver los agentes de A como controlables por el sistema y los restantes como elementos del entorno (incontrolable) subyacente. La alternancia aparece entonces puesto que el sistema puede ir controlando, vía elecciones (existenciales) de transiciones, el movimiento de sus agentes, al tiempo que estas se van exponiendo a las posibles elecciones complementarias del entorno, que por tanto deberán tratarse como elecciones no-deterministas cuantificadas universalmente. De este modo esa alternancia se traduce en cadenas de elecciones "existe"–"para todo".

La manera natural de relacionar ATS es por medio de las simulaciones alternantes [AHKV98] que generalizan la simulación convencional dentro de este formalismo de multi-agentes. Fijado un conjunto de agentes A, el juego de la simulación alternante partiendo (en particular) de los estados q y q', se puede resumir en el siguiente algoritmo de cuatro sencillos pasos extraído de [AHKV98].

1. El atacante elige un conjunto de movimientos T para los agentes de A desde q.

2. El defensor elige otro conjunto de movimientos T' para los agentes de A desde q'.

3. El atacante elige un estado $u' \in T'$ tal que sea sucesor de q'.

4. El defensor elige un estado $u \in T$ tal que sea sucesor de q y tal que el conjunto de proposiciones que se cumplen en u coincida con el conjunto de proposiciones que se cumplen en u'.

Como de costumbre tendremos que q' A-simula alternantemente a q si el defensor puede replicar constantemente cada movimiento del atacante manteniendo el equilibrio indicado en la cuarta cláusula del algoritmo anterior.

Los sistemas de transiciones tampoco incluyen ninguna información probabilística. Esta información se incluye en los denominados *sistemas de transiciones probabilísticos* (PTS del inglés *Probabilistic Labeled Transition Systems*) que fueron definidos por Larsen y Skou en 1991 [LS91] como tuplas $\mathcal{P}ts = (Pr, Act, Can, \mu)$, donde Pr es un conjunto de procesos, Act es el conjunto de acciones, Can indica el conjunto de procesos que pueden ejecutar la acción a y $\mu : Pr \times Act \times Pr \to [0, 1]$ es la función probabilística de transición, donde $\mu(P, a, \cdot)$ es o bien la función constante 0, o bien una distribución de probabilidad.

Es importante destacar que esta definición clásica de Larsen y Skou no nos permite hablar de "diferentes transiciones probabilísticas" que lleguen a un mismo proceso, es decir, una transición $P \xrightarrow{a}_{\mu} P'$ en cierto modo acumula toda las posibles maneras de ir desde p hasta p' tras ejecutar a.

La noción de bisimulación probabilística viene dada por la siguiente definición.

Definición 2.1.8 ([LS91])**.** *Sea $\mathcal{P} = (Pr, Act, Can, \mu)$ un sistema de transiciones probabilístico. Una bisimulación probabilística $\equiv$, es una relación de equivalencia en Pr tal que siempre que $p \equiv q$, se tiene*

$$\forall a \in Act.\, \forall S \in Pr/\equiv.\; p \xrightarrow{a}_{\mu} S \Leftrightarrow q \xrightarrow{a}_{\mu} S\ .$$

2.1.3. Estructuras con entrada y salida

Una de las carencias más importantes del formalismo de los sistemas de transiciones es que estos no permiten distinguir entre distintos tipos de etiquetas o acciones y por tanto clasificar sus acciones. Una correcta distinción de acciones es ya necesaria, por ejemplo, para modelizar correctamente un ejemplo tan sencillo como el de una máquina expendedora de café. Por ejemplo, la máquina puede tener una ranura para monedas, un botón y una bandeja; y su funcionamiento es tan elemental como aceptar una moneda, permitirnos presionar el botón, para después, dejar en la bandeja un café; acto seguido la máquina vuelve a su estado inicial. Para describir correctamente al completo esta máquina sería deseable tener algún modo de indicar que mientras que el comportamiento de las acciones *moneda* y *botón* necesita de la participación activa del usuario (ya que sin que nadie introduzca la *moneda* o apriete el *botón*, la máquina no realizará ninguna acción), el *café* lo deposita la máquina en la bandeja sin necesitar que delante de esta haya ningún usuario interviniendo.

La distinción que estamos haciendo en el ejemplo anterior es justamente una distinción entre acciones de entrada (*inputs*) y de salida (*outputs*). Las entradas se comportan como acciones reactivas, es decir, podemos pensar en ellas como botones que necesitan que algún agente externo los pulse para que el sistema avance o reaccione ante este impulso. En un sistema de transiciones estándar se entiende implícitamente que todas las acciones son reactivas. En cambio, una salida debe comportarse de un modo generativo, es decir, es el resultado del cómputo de la máquina, lo que la máquina "genera".

Las acciones de entrada facilitan más comportamientos posibles de una máquina; por ejemplo, nuestra máquina de cafés puede "mejorarse" incluyendo un segundo botón que permitiese pedir chocolate. Pero, de manera natural, las salidas introducen restricciones en el modelo. Siguiendo con el ejemplo de la máquina de café, ¿qué pasaría si en la máquina no existiese la bandeja en la que pueda depositar el café? Sin esa bandeja, o bien el café producido por la máquina caería al suelo (fastidiando así al usuario que pretendía tomárselo), o bien podríamos considerar que la propia máquina, al comprobar que no tiene ningún sitio dónde dejar el café, se bloquearía. Así, si añadiésemos a la máquina de café la opción de dar también chocolate, estamos imponiendo a la máquina que sea también capaz de tratar adecuadamente la salida *chocolate*, como por ejemplo, que el café y el chocolate no salgan por el mismo conducto para evitar que los sabores se mezclen.

Uno de los primeros modelos en incluir esta distinción entre acciones de entrada y de salida es el de los *autómatas con entrada/salida* o *I/O automata*. Este modelo se lo debemos a Nancy Lynch, que en 1988 publicó el artículo "I/O automata: a model for discrete event systems" [Lyn88]. Modelo que a lo largo de los años la propia Lynch ha ido extendiendo, por ejemplo, añadiéndoles tiempo (los *timed I/O automata* [KLSV03]), o tiempo y probabilidades (*probabilistic timed I/O automata* [CLSV04]).

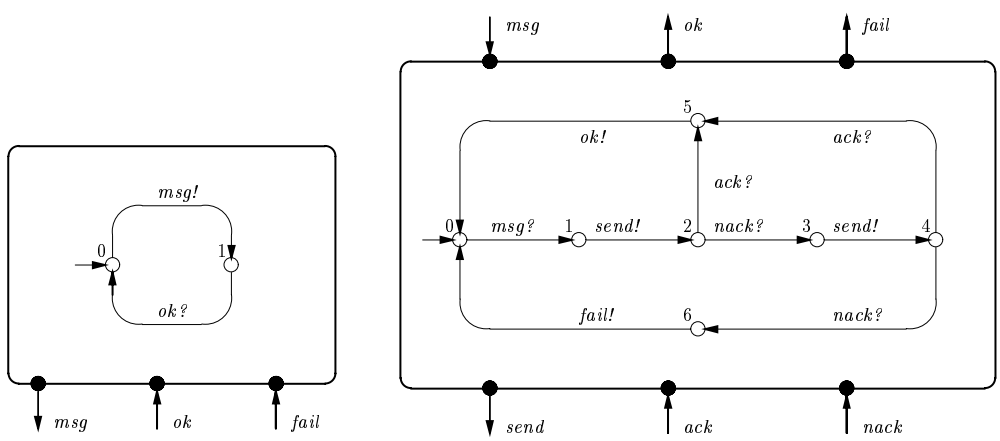
Lynch no siguió exactamente el modelo que presentábamos antes, pues distinguió no solo entre acciones de entrada y de salida, sino que también añadió un tercer tipo de acciones denominadas *internas*. Estas acciones internas, al igual que las acciones de salida, son generadas autónomamente por la máquina, pero no son transmitidas al exterior, como en el caso de las salidas. Como la propia autora comenta en [Lyn88], la distinción entre entrada y salida es primordial en los autómatas con entrada/salida. En particular, el formalismo impide cualquier bloqueo por culpa de la no atención a las acciones de entrada. A esto se conoce como *input enabled*, es decir, el sistema tiene que ser capaz de aceptar cualquier acción de entrada en cualquier momento, lo que no significa que el sistema no pueda tratar de manera especial aquellas entradas que, en un determinado momento de su ejecución, se consideren perjudiciales. Por ejemplo, ante una de esas entradas perjudiciales algunos de estos tratamientos especiales pueden ser desde generar un mensaje de error, hasta producir un comportamiento aleatorio.

El formalismo de Lynch permite también la composición de autómatas con entrada/salida, obteniéndose un marco composicional que sigue el estilo de la programación estructurada, de modo que se pueden unir distintos autómatas para construir un sistema con más funcionalidades. Para unir los autómatas es necesario que ambos sistemas sean compatibles, es decir, que las acciones de salida no lo sean en más de un autómata, y que las acciones de entrada e internas no aparezcan como cualquier otro tipo de acción en el resto de autómatas. La composición resultante une ciertas salidas de los autómatas

con ciertas entradas de los otros. De esta manera, el nuevo sistema actúa como un único autómata con entrada/salida, considerándose únicamente las entradas de los sistemas que no son también salida de alguno; y las salidas y acciones internas de ambos sistemas en conjunto. En cuanto al comportamiento de una composición, las acciones se ejecutan de manera concurrente en todas aquellas componentes que la tuviesen, mientras que las componentes que no la tuviesen se mantienen sin hacer nada.

Se podría decir que la manera en la que los autómatas con entrada/salida se componen sigue una visión "pesimista", pues considera que dos componentes se pueden componer únicamente si al juntarse no van a encontrar ninguna colisión y, por tanto, producir un error. En este contexto se considera que el ambiente con el que el autómata debe componerse es otro autómata con entrada/salida, de manera que podemos considerar que el formalismo de Lynch compone un autómata únicamente si el ambiente no tiene alguna manera de rechazarlo. En otras palabras, la composición de autómatas con entrada/salida únicamente es posible cuando entre ellos no existe forma alguna de llegar a una contradicción.

Existe la manera dual de interpretar la composición y es considerar que dos sistemas pueden componerse siempre que exista un ambiente bajo el que puedan funcionar juntos (esta es la visión "positiva"). Esta idea es la usada por Luca de Alfaro y Thomas Henzinger cuando propusieron en 2001 los *interface automata* [dAH01]. La semántica de los *interface automata* se define mediante un juego en el que un jugador maneja las entradas y representa el entorno, y el otro jugador maneja las salidas y representa al propio *interface automata*.



(a) Interface automaton *User*

(b) Interface automaton *Comp*

Figura 2.2: Dos *interface automata* tal y como aparecen en el artículo [dAH01].

Desde el punto de vista sintáctico (véase la figura 2.2), y como los mismos autores destacan en [dAH01], los *interface automata* tienen muchos puntos en común con los

autómatas con entrada/salida, e igualmente se distingue en ellos entre tres tipos de acciones: entrada, salida y acciones internas. Es el cambio de filosofía lo que marca la diferencia y, en particular, este cambio entraña la pérdida de la propiedad *input enabled*, lo que se traducirá en una manera diferente de componer autómatas. Así los *interface automata* no tienen porqué aceptar cualquier input en cualquier estado ya que la filosofía "optimista" entiende que el entorno no va a generarlos en ellos, como contraposición a la visión "pesimista" que considera que el entorno puede generar cualquier *entrada* en cualquier momento, por lo que el sistema debe mantenerse receptivo en cualquier estado.

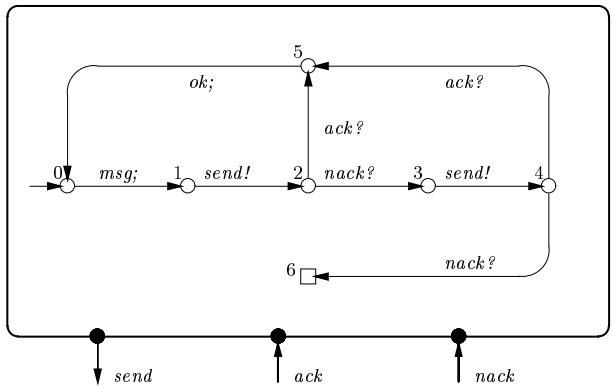
En cuanto a la composición de *interface automata*, al contrario de lo que pasaba en el caso de los autómatas con entrada/salida, aquí pueden aparecer estados *ilegales* en los que una acción de salida de uno de los *interface automata*, pese a ser acción de entrada del otro, no sea aceptada en ese estado. Pese a que existan estos estados, esto no significa que los *interface automata* no puedan componerse (véase la figura 2.3 y compárense las figuras (*c*) y (*d*)), pues dos *interface automata* son compatibles si existe un ambiente bajo el que puedan componerse. Por lo tanto, se puede argumentar que los *interface automata* son un marco más general que el de los autómatas con entrada/salida.

Es especialmente interesante hablar de la noción de *refinamiento* entre *interface automata*, que es la noción correspondiente en este formalismo al de simulación entre sistemas de transiciones, ya que nosotros hemos usado algunas de estas ideas, como se verá en el capítulo 3. En este contexto, se considera que un sistema P que refina a otro S, es una *implementación* de la *especificación* definida por S. En el caso de sistemas con *input enabled*, como destacan los propios autores de [dAH01], la noción más razonable de refinamiento es sencillamente la simulación, es decir, los comportamientos de la implementación deben estar incluidos en los comportamientos posibles dados por la especificación, ya que también garantiza que los comportamientos de las acciones de salida de la implementación son, a su vez, comportamientos que están permitidos por la especificación. En cambio, esta definición no parece razonable en un contexto como el de los *interface automata*, pues si se pide que el conjunto de entradas de la implementación sea un subconjunto de las entradas de la especificación, entonces, en general, una implementación se va a poder usar en menos entornos que la especificación.

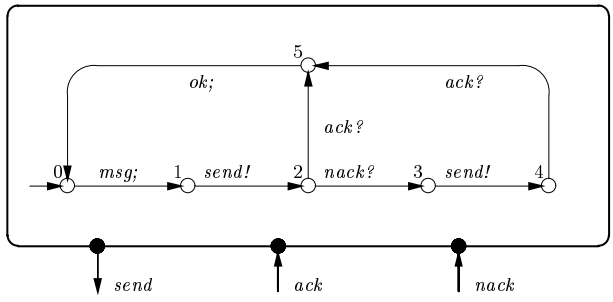
La propuesta de Alfaro y Henzinger es la de considerar un refinamiento de manera contravariante, es decir, en el sentido contrario al de la simulación estándar. De este modo, un *refinamiento entre interface automata* considera que la implementación debe permitir más entradas, mientras debe permitir menos salidas que la especificación. En realidad, los autores están considerando el refinamiento como un caso particular de la simulación alternante que vimos en la página 20; en el que el refinamiento debe tratar adecuadamente las acciones internas, es decir, actuar de un modo similar al de la *bisimulación débil* y considerar como posibles sucesores de un estado q todos aquellos estados q' a los que se puede llegar ejecutando solo acciones internas.

2.1.4. Sistemas modales y mixtos

Como hemos ido viendo en la sección 2.1.3, el formalismo de los LTS no era tampoco el ideal si lo que se deseaba era refinar una especificación o, en otras palabras, saber cuándo un LTS determinado puede considerarse una implementación de otro dado. La clave es



(c) *User* ⊗ *Comp*. The illegal state of the product is depicted as a square.



(d) *User*‖*Comp*

Figura 2.3: La composición de los *interface automata* de la figura 2.2, tal y como aparece en el artículo [dAH01].

que los sistemas de transiciones no permiten distinguir entre comportamientos posibles y comportamientos impuestos. Una transición $x \xrightarrow{a} y$ en un sistema de transiciones simplemente indica la *posibilidad* de que el estado x evolucione al estado y tras ejecutar a, pero no nos informa de si esa evolución es algo que se imponga o, simplemente, indica un comportamiento permitido.

A partir de esta sencilla idea, en 1988 Larsen y Thomsen definieron en [LT88] los *sistemas de transiciones modales* (abreviadamente MTS del inglés *modal transition systems*) o, en el caso en el que añadamos a cada estado conjuntos de proposiciones, las *especificaciones modales*. Los MTS distinguen entre dos tipos de transiciones: las requeridas o transiciones *must*, y las permitidas o transiciones *may*. Además, en los sistemas de transiciones modales, se considera que toda transición requerida es, en particular, una transición permitida. Formalmente, tenemos la siguiente definición (véanse también los

ejemplos de la figura 2.4):

Definición 2.1.9. *Dado un conjunto de acciones A, un* sistema de transiciones modal *es una terna $(P, \longrightarrow_\diamond, \longrightarrow_\Box)$, donde P es el conjunto de estados y $\longrightarrow_\diamond, \longrightarrow_\Box \subseteq P \times A \times P$ son relaciones de transición tales que $\longrightarrow_\Box \subseteq \longrightarrow_\diamond$.*

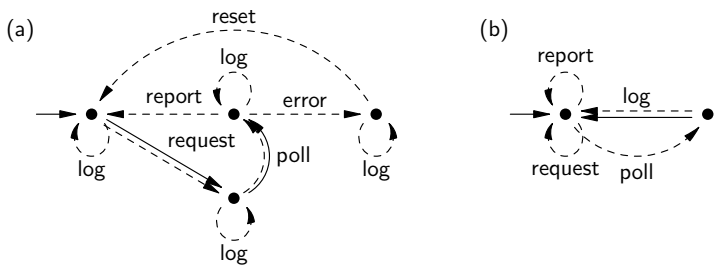


Figura 2.4: Dos MTS sacados de [AHL$^+$08]. Las transiciones *may* están representadas por transiciones punteadas, mientras que las *must* lo están por transiciones continuas.

El papel que desempeñan en los sistemas de transiciones modales las transiciones *may* y *must* se observa estudiando la noción de refinamiento propuesta por Larsen y Thomsen en [LT88]. Intuitivamente, un MTS N refina a otro MTS M siempre que todas las transiciones requeridas por M están también presentes en N, y toda transición que tenga N está permitida en M. Formalmente, la definición es la siguiente:

Definición 2.1.10. *Una relación $R \subseteq P \times Q$ es un* refinamiento *entre dos sistemas de transiciones modales si, siempre que $p\ R\ q$:*

- *$p \xrightarrow{a}_\Box p'$ implica que existe un q' tal que $q \xrightarrow{a}_\Box q'$ y $p'\ R\ q'$;*
- *$q \xrightarrow{a}_\diamond q'$ implica que existe un p' tal que $p \xrightarrow{a}_\diamond p'$ y $p'\ R\ q'$.*

De este modo, si N refina a M, entonces N puede transformar en transiciones *must* algunas transiciones *may* de M, o bien eliminar alguna de ellas. Así, según se va refinando un sistema de transiciones modal, cada vez hay más transiciones "completamente definidas", es decir, más transiciones *must*. Siguiendo la terminología de Larsen y Thomsen, cuando en un MTS todas las transiciones son *must*, entonces se considera que estamos ante un *proceso* o, sencillamente, una *implementación*, mientras que en otro caso el MTS representa un *término* o una *especificación*.

Evidentemente existen ciertos nexos en común entre los sistemas de transiciones modales y sus refinamientos, y los *interface automata* con sus refinamientos y, por lo tanto, con la simulación alternante. Este "parecido razonable" entre ambas teorías proviene en parte de que en ambos marcos existen ciertas aspectos que son controlables, mientras que otros aspectos escapan a dicho control. En el caso de los *interface automata* el refinamiento nos permite eliminar acciones de salida (el aspecto "controlable"), pero no de

entrada (el aspecto "incontrolable"); mientras que en el modelo de los sistemas de transiciones modales, el refinamiento nos permite eliminar transiciones *may* (controlables) y no transiciones *must*.

No obstante, Larsen, junto con otros autores, conjeturan en el artículo [LNW07] que las simulaciones alternantes son un caso particular de los refinamientos modales. Esta afirmación la justifican construyendo una transformación que convierte un *interface automaton* en un sistema de transiciones modal. Esta transformación, además, refleja y preserva las respectivas simulaciones y refinamientos.

Por otro lado, y como se hace notar en [RBB+09], se puede argüir que los modelos son incomparables pues, en cierto sentido, el carácter de las acciones de entrada y salida es ortogonal al uso de las modalidades *may* y *must*. Además, ambos formalismos tienen sus ventajas e inconvenientes pues, por ejemplo, el formalismo de los *interface automata* permite definir una noción sencilla e intuitiva de composición, lo que lo hace tremendamente útil a la hora de trabajar modularmente, pero a su vez estos no forman un modelo algebraico completo, al carecer de operaciones tan básicas como el cociente o la conjunción. En cambio, los sistemas de transiciones modales no permiten una noción de composición intuitiva, pero sí dan lugar a un modelo algebraico muy rico.

Apoyándose en estas diferencias entre los dos modelos, en ese mismo artículo de 2007 [LNW07], Larsen y el resto de autores proponen un modelo que aúna a los *interface automata* y los MTS, denominado *modal I/O automata* que, esencialmente, es un *interface automaton* que permite transiciones *may* y *must*, de acuerdo con el modelo de los MTS.

Dentro del estudio de los MTS, el artículo de Boudol y Larsen [BL92] es de obligada referencia. En él, los autores estudian la relación entre los procesos modales y sus fórmulas. Para ello, construyen tanto una caracterización recursiva de las fórmulas características de los procesos, como indican los procesos que representan a una fórmula dada.

La lógica modal que caracteriza el refinamiento es muy parecida a la lógica de Hennessy-Milner de la bisimulación. Su sintaxis viene dada por la siguiente gramática

$$\varphi ::= \bot \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [a]\varphi \mid \langle a \rangle \varphi \qquad (a \in A).$$

Pero si bien su semántica es la estándar para los operadores clásicos, no lo es para los operadores modales $[a]$ y $\langle a \rangle$ que quedan definidos así:

- $p \models [a]\varphi$ si $p' \models \varphi$, para todo $p \xrightarrow{a}_{\diamond} p'$.
- $p \models \langle a \rangle \varphi$ si $p' \models \varphi$ para algún $p \xrightarrow{a}_{\Box} p'$.

Como en particular toda transición *must* es también *may*, para comprobar que una fórmula $[a]\varphi$ es cierta puede usarse también una transición *must*.

Por otro lado, una fórmula ϕ es una *fórmula característica* de un proceso p si, y solo si, $p \models \phi$ y para todo proceso q tal que $q \models \phi$ se tiene que q es un refinamiento de p.

En 1996 Dams [Dam96, DGG97] definió una variante de los sistemas de transiciones modales denominados *sistemas de transiciones mixtos* (o abreviadamente MiTS), con la única diferencia con respecto a los MTS de que en los MiTS no se tiene la inclusión

$\longrightarrow_{\Box} \subseteq \longrightarrow_{\diamond}$. De esta manera, los MiTS generalizan a los MTS y también pueden verse como LTS con dos tipos de transiciones distintas.

Como se comenta en [AHL$^+$08] la violación de que no toda transición requerida está permitida, pese a ser intuitivamente extraña, no es tan descabellada. En primer lugar, en muchos casos es deseable poder expresar en una especificación comportamientos conflictivos, que se podrían considerar contradictorios al no existir una implementación para ella. Pero todo MTS admite una implementación: basta quedarse con la información proporcionada por las transiciones *must* y descartar todas aquellas transiciones *may* que no viniesen de otra *must*, como se recoge en la figura 2.5. Por el contrario, un MiTS puede expresar la situación de contradicción al poder decir que una acción es requerida pero no está permitida, es decir, se tiene una transición *must* que no es también *may*. En segundo lugar, cuando la especificación es la abstracción de un programa puede ser deseable mejorar esa abstracción incluyendo más propiedades que se satisfacen, y esto se puede hacer [Dam96, DGG97] eliminando ciertas transiciones *may* redundantes y añadiendo nuevas transiciones *must* (no relacionadas con ninguna *may*).

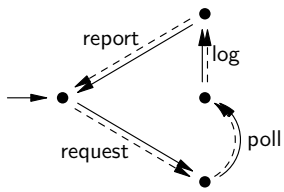


Figura 2.5: Una implementación común de los MTS de la figura 2.4, también extraído de [AHL$^+$08].

Como hemos comentado en la sección 1.2, existen otras muchas extensiones de los MTS tal y como se recapitulan en [AHL$^+$08], como pueden ser los *sistemas de transiciones modales disyuntivos* [LX90], en los que existen transiciones *must disyuntivas* que especifican que la implementación debe implementar al menos una de sus disyunciones, o extensiones de los MTS con tiempo [CGL93].

2.2. Teoría de categorías y coálgebras

Una pregunta que todo investigador que haya decidido emprender el estudio de las coálgebras se habrá hecho, y le habrán planteado más de una vez, es: ¿por qué usar coálgebras para trabajar con sistemas de transiciones?

Ante tal pregunta existen varias respuestas y, en primer lugar, podemos defendernos diciendo que las coálgebras son tremendamente generales y permiten representar una gran variedad de sistemas de un modo elegante, como veremos a lo largo de la presente sección. En segundo lugar, al tener las coálgebras detrás las herramientas de la teoría de categorías, también se puede argüir que las coálgebras son una herramienta relativamente sencilla en tanto y cuanto, para familiarizarse con ellas, basta apoyarse únicamente en

una serie de conceptos básicos de la teoría de categorías. Pero quizá la más importante razón por la que creemos que es especialmente interesante usar coálgebras para estudiar las álgebras de procesos, es que se puede decir que las coálgebras nos permiten ver a las álgebras de procesos del modo más natural. En lo que resta de sección vamos a motivar esta última afirmación.

Hablábamos en la sección 1.3 de cómo en los años 70 Goguen y el resto de sus coautores usaron la teoría de categorías para asentar las bases de la especificación algebraica y, con ella, los tipos abstractos de datos. Goguen vio que a la hora de definir un tipo de datos era de especial importancia el concepto de álgebra inicial. Esta *inicialidad* es la que nos permite caracterizar la sintaxis de nuestro tipo de datos, mientras que las posibles semánticas vienen dadas por las álgebras de la clase, viniendo definidas por el único homomorfismo desde el álgebra inicial. Pero la inicialidad de las álgebras tiene también otra lectura (aunque en el fondo se trate de una relectura del anterior) que ve esa existencia y unicidad del homomorfismo como el *principio de inducción*: la existencia es la definición por inducción, mientras que la unicidad es la prueba por inducción (dadas dos funciones actuando sobre un álgebra inicial se prueba que son la misma viendo que ambos son homomorfismos).

El concepto dual (para las coálgebras) de la inicialidad no es otro que el de finalidad, como ya vio Aczel en [Acz88]. Así, si decíamos que se podía ver un álgebra inicial como el álgebra más pequeña de una determinada clase, la *coálgebra final* sería por el contrario la mayor, ya que dada una coálgebra arbitraria siempre existe un homomorfismo de esta hacia la coálgebra final (la flecha contraria a la que aparecía en la inicialidad). Asimismo, mientras que el principio de razonamiento lógico en las álgebras es la inducción, en el marco dual de las coálgebras tendremos la *coinducción*, de la cual es una instancia la bisimulación (como haremos más preciso en la sección 2.2.2). Por otro lado, siguiendo con el símil de los tipos abstractos de datos, si estos están generados por los operadores constructores, dualmente, en el mundo coinductivo, serán las operaciones denominadas observadoras o modificadoras, y no las constructoras, las que realizarán una función principal.

Para ilustrar este hecho veamos un ejemplo. Pensemos en la especificación algebraica clásica de las listas de elementos de un conjunto A fijo, que como es usual denotaremos por A^*. Tenemos dos operaciones constructoras:

- $\langle\rangle : \mathsf{nil} \to A^*$: es el constructor constante que permite crear la lista vacía.
- $\cdot : A \times A^* \to A^*$ es el operador prefijo que permite la adición del prefijo $a \in A$ a la lista $l \in A^*$ para formar la nueva lista $a \cdot l$.

Junto con estas dos constructoras podemos añadir dos operaciones observadoras (o destructoras): la que dada una lista devuelve su cabeza (es decir, el primer elemento de ella; nótese que, por tanto, la lista no puede ser vacía) y la que devuelve su cola (es decir, toda la lista salvo la cabeza). Definimos así la función parcial $cabeza : A^* \to_P A$ y la función $cola : A^* \to A^*$ mediante inducción estructural:

- $cabeza(a \cdot l) = a$.
- $cola(\langle\rangle) = \langle\rangle$.

- $cola(a \cdot l) = l$.

Con esta especificación algebraica estamos definiendo listas que son finitas, ya que estamos usando el principio de inducción de las álgebras, que nos dice que solo existen aquellos elementos que pueden obtenerse mediante aplicación de un número finito de los constructores. Pero, ¿qué pasa si lo que queremos definir es el tipo de datos de las listas infinitas? En tal caso deberemos recurrir a la teoría coalgebraica y realizar una definición coinductiva. La principal diferencia entre una definición inductiva y otra coinductiva es que en las definiciones inductivas se define el valor de la función dada actuando sobre cada una de las constructoras, mientras que las definiciones coinductivas definen el valor de todas las observaciones sobre cada posible valor de la función.

Por ejemplo: dada una función f que lleve elementos de un conjunto A sobre él mismo vamos a definir una extensión suya, que denotaremos por $ext(f)$, que lleve listas infinitas de elementos de A sobre ellas mismas. Para ello tendremos que dar una definición coalgebraica y por tanto, como dijimos antes, tendremos que dar el valor de las observadoras *cabeza* y *cola* sobre cada secuencia $ext(f)(l)$. En concreto tendremos:

$$\begin{cases} cabeza(ext(f)(l)) & = & f(cabeza(l)) \\ cola(ext(f)(l)) & = & ext(f)(cola(l)) \,. \end{cases}$$

Así, por ejemplo podemos definir dos funciones *par* e *impar* que tomen listas infinitas y devuelvan las listas (también infinitas) formadas por los elementos que se encontraban en las posiciones pares e impares, respectivamente:

$$\begin{cases} cabeza(impar(l)) & = & cabeza(l) \\ cola(impar(l)) & = & impar(cola(cola(l))) \end{cases}$$

$$\begin{cases} cabeza(par(l)) & = & cabeza(cola(l)) \\ cola(par(l)) & = & par(cola(cola(l))) \,. \end{cases}$$

Como se ve al trabajar con definiciones inductivas construimos las listas finitas, mientras que al considerar una definición coinductiva vemos cómo se comportan las listas infinitas sin necesidad de construir estas de un modo explícito. Esta filosofía es especialmente útil a la hora de capturar el comportamiento de las *clases* de los lenguajes de programación orientados a objetos, como pueden ser `Java` o `C++`. Como se indica en [JR97] si, por ejemplo, queremos capturar los aspectos esenciales de la clase `Point` de puntos en el plano real, tendremos dos métodos observadores `getX`: $P \to \mathbb{R}$ y `getY`: $P \to \mathbb{R}$, que nos devolverán, respectivamente, la primera y segunda coordenada de cada punto. Aquí la P juega el papel de un espacio de estados oculto, es decir, la P se identifica con los objetos de la clase `Point` que, como suele suceder en la programación orientada a objetos, no tiene porqué ser "abierta" a cualquier usuario.

Podemos también definir un nuevo método `move` : $P \times \mathbb{R} \times \mathbb{R} \to P$ que dados dos parámetros reales modifica las coordenadas del punto. Como decíamos antes, el cliente de este objeto no tiene porqué estar interesado en la implementación concreta de la clase `Point`, ni la de sus métodos privados, pero sí que puede especificar el comportamiento del método `move` con las siguientes ecuaciones:

- $s.\texttt{move}(d1, d2).\texttt{getX}() = s.\texttt{getX}() + d1$
- $s.\texttt{move}(d1, d2).\texttt{getY}() = s.\texttt{getY}() + d2$

Damos ahora la definición (casi) formal de coálgebra. Decimos que una coálgebra es una función:

$$S \xrightarrow{c} TS\,.$$

Es decir, una coálgebra viene dada por un conjunto S y una función c, con S como dominio y cuyo codominio es TS. Usamos la notación TS para indicar que se trata de una expresión dependiente del conjunto S, que más adelante (en concreto en la página 35) veremos exactamente en qué consiste.

Veamos ahora otro ejemplo extraído de [JR97]. Imaginemos que tenemos una máquina con un único botón y una luz. Cuando se aprieta el botón la máquina realiza algún tipo de acción (posiblemente exista un estado interno que cambie cada vez que se pulse el botón, pero que nosotros no vemos). La luz se encenderá únicamente si la máquina se rompe; con ello, una vez que la luz se encienda da igual la cantidad de veces que pulsemos el botón, porque nada cambiará ya nunca. Un observador puede contar la cantidad de veces que se pulsa el botón hasta que la máquina se rompe, y ese número puede ser tanto un cierto $n \in \mathbb{N}$ como una cantidad infinita.

Desde un punto de vista matemático se puede describir el funcionamiento de la máquina en términos de un conjunto X desconocido y una función

$$\mathit{botón} : X \to \{*\} \cup X\,,$$

donde el símbolo $*$ es un símbolo especial que no aparece en X, y que indicará el momento en que la luz se enciende. Por ejemplo, si la máquina está en un estado $x_0 \in X$ y pulsamos el botón pasaremos a un nuevo estado $x_1 = \mathit{botón}(x_0)$; si $x_1 \neq *$ podremos volver a pulsar el botón, y así sucesivamente.

El par $(X, \mathit{botón})$ es un primer ejemplo de coálgebra. Sobre este sencillo modelo podemos construir otro más interesante: el similar al ya avanzado antes de las listas (posiblemente infinitas) de elementos de un conjunto A. Para ello, consideremos ahora una máquina con dos botones, una luz y un visor, que mostrará la información del estado en el que se encuentra la máquina. Como antes, uno de los botones servirá para pasar al siguiente estado, mientras que el otro iluminará en el visor algún signo que identifique el estado interno de la máquina. Si la luz se enciende significará que la máquina está rota. ¿Qué tenemos entonces? Pues exactamente lo siguiente:

$$\mathit{valor\_sig} : X \to \{*\} \cup (A \times X)$$

Como se observa, la función $\mathit{valor\_sig}$ (casi) coincide con el par de funciones que veíamos antes (el par $(\mathit{cabeza}, \mathit{cola})$), con la salvedad de que $\mathit{valor\_sig}$ permite la aparición de listas finitas (cuando la máquina se rompe). Así, cuando el observador se encuentra con esta máquina pueden suceder dos cosas: que la luz esté encendida o que esté apagada. En el primer caso la máquina está rota y, como en el ejemplo anterior, da igual las veces que pulsemos el botón sig, porque no cambiará nada. Por el contrario, en el segundo caso, el observador podrá pulsar reiteradamente en secuencia los botones valor y sig (de

ahí la notación usada para el nombre de la función haciendo explícitos los nombres de los botones) para obtener por el visor una lista de valores $a_i \in A$. De nuevo, tras una serie de pulsaciones, digamos n, la luz puede encenderse, en cuyo caso obtendremos una lista finita $(a_0, \ldots, a_n)$, mientras que si no llega a encenderse nunca obtendremos una lista infinita de elementos (a_i), $i \in \mathbb{N}$. El par $(X, valor\_sig)$ es una coálgebra.

Volviendo ahora al ejemplo de la clase `Point`, en primer lugar podemos redefinir equivalentemente el tipo del método `move` : $P \times \mathbb{R} \times \mathbb{R} \to P$ como `move` : $P \to P^{\mathbb{R}\times\mathbb{R}}$. De este modo podemos considerar la coálgebra:

$$(\texttt{getX}, \texttt{getY}, \texttt{move}) : \mathbb{R} \times \mathbb{R} \times P^{\mathbb{R}\times\mathbb{R}}$$

Esta representa una máquina que tiene tres botones y dos visores. Dos de los botones iluminan cada uno un visor, en el que se verá el resultado de hacer `getX` y `getY`, mientras que el tercer botón sirve para avanzar el estado según las ecuaciones que indicamos antes que cumple `move`.

Decíamos al inicio de esta sección que existía una tercera razón por la que nos habíamos decantado por el uso de las coálgebras, y que esta razón era que nos permitía trabajar con las álgebras de procesos de un modo muy natural. Esta naturalidad está fundamentada en varios aspectos esenciales que hemos ido comentando. Hemos dicho que la manera natural de definir coálgebras es usando la coinducción que, como veremos, está íntimamente relacionado con la bisimulación. Además, está ya ampliamente justificada que la bisimulación es la manera natural y canónica de comparar procesos. A esto podemos añadir el hecho de que las definiciones coinductivas nos permiten trabajar con estructuras infinitas de un modo muy sencillo, al margen de la farragosidad que a veces introducen los puntos fijos o la inducción transfinita. Además nos permiten capturar las nociones de clase privada o *caja negra* (que veíamos en la página 3), en las que únicamente podemos observar cierta información facilitada por la clase o máquina, pero no conocer ni su configuración interna ni cómo funciona exactamente. Es decir, al trabajar con coálgebras tenemos "gratis" un formalismo que captura las propiedades básicas deseables de los sistemas informáticos.

Más aún, el comportamiento de un proceso P viene dado por sus transiciones, es decir, podemos decir que su semántica viene dada por: $[\![P]\!] = \{\langle a, [\![P_i]\!]\rangle \mid P \xrightarrow{a} P_i\}$. De esta manera podemos ir observando las acciones que el proceso va ejecutando, y al hacerlo este, se irá transformando en sucesivos P'. El problema es que podemos encontrarnos con ciclos que a partir de P terminan devolviendonos a P, por lo que el uso de la inducción no nos permitirá definir con propiedad el conjunto $[\![P]\!]$. Es decir, si vemos los sucesores como "partes" del proceso original nos encontraríamos con una cadena supuestamente decreciente

$$\ldots \in P_k \in \ldots \in P_2 \in P_1 \in P_0 = P$$

en la que de pronto saltarían las alarmas al encontrarnos con cierto P_i con $i \geq 0$ igual a P.

Si bien las técnicas de punto fijo utilizadas en semántica denotacional permiten abordar este problema introduciendo un orden en el dominio semántico y obteniendo $[\![P]\!]$ como un límite, esta solución parece mucho menos natural que la que tomó Aczel cuando formuló el principio de anti-fundación [Acz88] y consideró que en CCS se trabajaba con

el modelo $[\![\cdot]\!]_{ccs}$, que podía verse como una coálgebra $[\![\cdot]\!]_{ccs} : Pr \to \mathcal{P}_\omega(Act \times Pr)$, donde Pr es el conjunto de procesos, Act el alfabeto de acciones y $\mathcal{P}_\omega(X)$ son las partes finitas del conjunto X. Así, la semántica $[\![P]\!]$ que definíamos antes no es otra cosa que ese único homomorfismo que existe desde esta coálgebra $[\![\cdot]\!]_{ccs}$ hacia la coálgebra final (o, como veremos en la sección 2.2.2, el mayor punto fijo) para la estructura $\mathcal{P}_\omega(Act \times X)$.

Por todo ello quizá la pregunta que nos hacíamos al principio de esta sección deberíamos reformularla a la inversa (o a la "dual"): si vamos a trabajar con álgebras de procesos, ¿por qué no usar coálgebras?

2.2.1. Categorías y coálgebras

Para formalizar la noción de coálgebra necesitaremos una serie de definiciones y conceptos del campo de la teoría de categorías (véanse, por ejemplo, [BW99, Mac98]), que nos van a resultar indispensables en nuestro tratamiento.

Empezaremos por la propia noción de categoría, recogida también en su enunciado original en la figura 2.6:

Definición 2.2.1 (Categoría)**.** *Una categoría* $\mathbb{C}$ *es una colección* $\mathsf{Obj}(\mathbb{C})$ *de* objetos*, una colección* $\mathsf{Arr}(\mathbb{C})$ *de* flechas *o* morfismos*, y un par de flechas* dom *y* codom *desde* $\mathsf{Arr}(\mathbb{C})$ *a* $\mathsf{Obj}(\mathbb{C})$ *denominadas respectivamente* dominio *y* codominio*.*

Generalmente utilizaremos la notación $X \in \mathbb{C}$ *(resp.* $f \in \mathbb{C}$*) para referirnos a los objetos* $X \in \mathsf{Obj}(\mathbb{C})$ *(resp. a los morfismos* $f \in \mathsf{Arr}(\mathbb{C})$*).*

Cada flecha de $\mathbb{C}$ *la denotaremos por* $X \xrightarrow{f} Y$ *o* $f : X \to Y$*, donde diremos que* $X \in \mathbb{C}$ *es el dominio de* f *(es decir,* $\mathsf{dom}(f) = X$*) e* $Y \in \mathbb{C}$ *el codominio de* f *(*$\mathsf{codom}(f) = Y$*).*

Además se cumplirán las siguientes condiciones:

1. *Para cada par de flechas* $f : X \to Y$ *y* $g : Y \to Z$ *existe una flecha composición* $g \circ f : X \to Z$*. Además, la operación* $\circ$ *es asociativa, es decir, si tenemos además* $h : Z \to W$ *entonces* $h \circ (g \circ f) = (h \circ g) \circ f$*.*

2. *Para cada objeto* $X \in \mathbb{C}$ *existe una flecha identidad* $id_X : X \to X$ *que es el elemento neutro de la composición* $\circ$*. Es decir, si* $f : X \to Y$ *entonces* $f \circ id_X = f = id_Y \circ f$*. Generalmente se omitirá el subíndice* X *de* id_X*, siempre que quede claro por el contexto.*

Una de las categorías más usadas en la literatura, así como la principal que hemos manejado a lo largo de la investigación que conforma esta tesis, es la categoría **Sets**, definida como la categoría que tiene por objetos los conjuntos y por morfismos las funciones ordinarias entre conjuntos. Si no lo indicamos expresamente, todas nuestras construcciones estarán en la categoría **Sets**.

Ejemplo 2.2.1 Veamos ahora algunos otros ejemplos de categorías:

1. Todo preorden $(D, \leq)$ es una categoría. Sus objetos son los elementos $d \in D$ y existe una flecha $d_1 \to d_2$ si y solo si $d_1 \leq d_2$.

2. Un monoide $(M, +, 0)$ también puede ser visto como una categoría con un único elemento $\sharp$ y una flecha $\sharp \to \sharp$ para cada elemento $m \in M$. Así, dados $m_1 : \sharp \to \sharp$ y $m_2 : \sharp \to \sharp$, la composición será $m_1 + m_2 : \sharp \to \sharp$, y 0 es la identidad.

3. **Grp**: es la categoría de los grupos en la que los objetos son grupos y las flechas son los homomorfismos (que preservan las composiciones y el elemento unidad) entre grupos.

4. **PreOrd**: es la categoría de los preórdenes (no confundir con el primer ejemplo, en el que veíamos un único preorden como una categoría) donde los objetos son preórdenes y las flechas son las funciones monótonas (es decir, aquellas que preservan el orden entre elementos) entre preórdenes.

A *category* $\mathfrak{A} = \{A, \alpha\}$ is an aggregate of abstract elements A (for example, groups), called the *objects* of the category, and abstract elements α (for example, homomorphisms), called *mappings* of the category. Certain pairs of mappings $\alpha_1, \alpha_2 \in \mathfrak{A}$ determine uniquely a product mapping $\alpha = \alpha_2\alpha_1 \in \mathfrak{A}$, subject to the axioms C1, C2, C3 below. Corresponding to each object $A \in \mathfrak{A}$ there is a unique mapping, denoted by e_A or by $e(A)$, and subject to the axioms C4 and C5.

C1. *The triple product* $\alpha_3(\alpha_2\alpha_1)$ *is defined if and only if* $(\alpha_3\alpha_2)\alpha_1$ *is defined. When either is defined, the associative law*

$$\alpha_3(\alpha_2\alpha_1) = (\alpha_3\alpha_2)\alpha_1$$

holds. This triple product will be written as $\alpha_3\alpha_2\alpha_1$.

C2. *The triple product* $\alpha_3\alpha_2\alpha_1$ *is defined whenever both products* $\alpha_3\alpha_2$ *and* $\alpha_2\alpha_1$ *are defined.*

DEFINITION. A mapping $e \in \mathfrak{A}$ will be called an *identity* of $\mathfrak{A}$ if and only if the existence of any product $e\alpha$ or βe implies that $e\alpha = \alpha$ and $\beta e = \beta$.

C3. *For each mapping* $\alpha \in \mathfrak{A}$ *there is at least one identity* $e_1 \in \mathfrak{A}$ *such that* αe_1 *is defined, and at least one identity* $e_2 \in \mathfrak{A}$ *such that* $e_2\alpha$ *is defined.*

C4. *The mapping* e_A *corresponding to each object* A *is an identity.*

C5. *For each identity* e *of* $\mathfrak{A}$ *there is a unique object* A *of* $\mathfrak{A}$ *such that* $e_A = e$.

Figura 2.6: La definición original de categoría, tal y como aparece en [EM45].

A aquellas categorías cuyos objetos y morfismos forman un conjunto se las suele denominar *pequeñas* (*small* en inglés), mientras que aquellas en las que no lo forman son denominadas *grandes* (en inglés *large*). La categoría **Sets** es una *categoría grande*[2].

[2]Como Bertrand Russell probó en 1901 (véase por ejemplo en el libro recopilatorio de Godehard Link [Lin04]), no existe "el conjunto formado por todos los conjuntos". A la hora de trabajar con una

Otro interesante ejemplo de categoría es la *categoría opuesta*, o *dual*, a otra dada. Esencialmente se trata de la misma categoría, pero donde las flechas se toman al revés, es decir, dada una categoría $\mathbb{C}$, su categoría opuesta, $\mathbb{C}^{op}$, es la categoría con objetos y morfismos definidos como sigue:

1. $X \in \mathrm{Obj}(\mathbb{C}^{op})$ si $X \in \mathrm{Obj}(\mathbb{C})$.
2. $f \in \mathrm{Arr}(\mathbb{C}^{op})$ con $f : Y \to X$, si $X \to Y \in \mathrm{Arr}(\mathbb{C})$.

Si bien ya hemos introducido antes los conceptos de inicialidad y finalidad, ahora una vez definido lo que es una categoría podemos formalizar con precisión qué es un objeto final y qué es un objeto inicial, en una categoría.

Definición 2.2.2 (Objeto inicial y final)**.** *Dada una categoría $\mathbb{C}$ diremos que:*

1. *Un objeto final es un objeto $1 \in \mathbb{C}$ tal que para cualquier otro objeto $X \in \mathbb{C}$ existe un único morfismo $!_X : X \to 1$.*
2. *Un objeto inicial es un objeto $0 \in \mathbb{C}$ tal que para cualquier otro objeto $X \in \mathbb{C}$ existe un único morfismo $!_X : 0 \to X$.*

No todas las categorías tienen objetos finales o iniciales pero, si existen, entonces son únicos, salvo isomorfismo. Aprovechamos para decir que un isomorfismo es un morfismo $f : X \to Y$ para el que existe otro morfismo $f^{-1} : Y \to X$, denominado *morfismo inverso*, tal que $f^{-1} \circ f = id_X$ y $f \circ f^{-1} = id_Y$.

Decíamos antes que una coálgebra no era más que una función $c : S \to TS$, donde TS dependía de S. Para precisar exactamente de qué modo TS depende de S necesitamos introducir la noción de *funtor*, es decir, la noción que nos permite precisar qué se entiende por una "función" entre categorías.

Definición 2.2.3 (Funtor)**.** *Dadas dos categorías $\mathbb{C}$ y $\mathbb{D}$, un funtor $F : \mathbb{C} \to \mathbb{D}$ viene dado por dos funciones $\mathrm{Obj}(\mathbb{C}) \to \mathrm{Obj}(\mathbb{D})$ y $\mathrm{Arr}(\mathbb{C}) \to \mathrm{Arr}(\mathbb{D})$ (denotadas generalmente ambas por F) tales que:*

1. *F preserva dominios y codominios. Así dado $f : X \to Y$ en la categoría $\mathbb{C}$ entonces $F(f) : F(X) \to F(Y)$ en la categoría $\mathbb{D}$.*
2. *F preserva las identidades. Así, para cada $X \in \mathbb{C}$, $F(id_X) = id_{F(X)}$.*
3. *F preserva la composición. Para todo par de flechas $f : X \to Y$ y $g : Y \to Z$ en la categoría $\mathbb{C}$, se tiene que $F(g \circ f) = F(g) \circ F(f)$.*

Los funtores $F : \mathbb{C} \to \mathbb{C}$ de una categoría en sí misma, se denominan endofuntores.

Para cada categoría $\mathbb{C}$ existe siempre un funtor identidad trivial $id_{\mathbb{C}} : \mathbb{C} \to \mathbb{C}$ que lleva f a f y X a X. También, para cada objeto $A \in \mathbb{C}$ se puede definir un funtor constante $A : \mathbb{D} \to \mathbb{C}$ que a cada objeto $Y \in \mathbb{D}$ le hace corresponder el objeto $A \in \mathbb{C}$ y a cada morfismo $f \in \mathbb{D}$ le hace corresponder la identidad $id_A : A \to A$.

estructura que permita eludir las paradojas de la teoría de conjuntos se suele trabajar con la categoría de las **Clases**.

Además, dados dos funtores $F : \mathbb{C} \to \mathbb{D}$ y $G : \mathbb{D} \to \mathbb{E}$, existe el funtor composición definido por

$$G \circ F : \mathbb{C} \to \mathbb{E}, \quad \text{con} \quad X \mapsto G(F(X)) \quad \text{y} \quad f \mapsto G(F(f)) .$$

Generalmente, y siempre que ello no genere confusión, se suelen evitar los paréntesis al trabajar con funtores, y así FX denotará $F(X)$.

Ejemplo 2.2.2 Veamos ahora algunos ejemplos sencillos de funtores interesantes:

1. Dados dos monoides $(M, +, 0)$ y $(N, \cdot, 1)$ en la categoría de los monoides, un funtor $F : M \to N$ es un homomorfismo entre monoides, es decir, una función que preserve los operadores de composición y el elemento neutro.

2. Análogamente, si consideramos dos preórdenes $(D, \leq)$ y $(E, \sqsubseteq)$ como categorías, un funtor $F : D \to E$ no es otra cosa que una función monótona, es decir, tal que si $x \leq y$ entonces $F(x) \sqsubseteq F(y)$.

3. Un ejemplo aparentemente trivial, pero muy útil, lo constituyen los funtores olvidadizos (en inglés *forgetful functor*) que, como su nombre indica, olvidan parte de la estructura de su dominio. Por ejemplo entre **PreOrd** $\to$ **Sets** existe un funtor que "olvida" la estructura del los preórdenes $(P, \sqsubseteq)$ quedándose solo con el conjunto subyacente P.

Otro concepto importante dentro del universo categórico es el de *transformación natural* que, esencialmente, no es sino un morfismo entre funtores, que pasamos a definir con absoluta propiedad a continuación.

Definición 2.2.4 (Transformación natural)**.** *Sean $\mathbb{C}$ y $\mathbb{D}$ dos categorías y F y G dos funtores de $\mathbb{C}$ en $\mathbb{D}$. Una transformación natural α de F a G es una colección de flechas $\alpha_X : FX \to GX$ en $\mathbb{D}$ para cada objeto $X \in \mathbb{C}$ que cumplen que para cada flecha $f : X \to Y$ en $\mathbb{C}$, el siguiente diagrama es conmutativo, lo que se conoce como condición de naturalidad:*

$$\begin{array}{ccc} FX & \xrightarrow{\alpha_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\alpha_Y]{} & GY \end{array}$$

Genéricamente se denotará por $\alpha : F \Rightarrow G$ a una transformación natural.

Así, ya estamos en disposición de dar la definición formal de coálgebra.

Definición 2.2.5 (Coálgebra)**.** *Sea $\mathbb{C}$ una categoría cualquiera y F un endofuntor de $\mathbb{C}$ en $\mathbb{C}$.*

1. *Una F-coálgebra o, simplemente, una coálgebra (cuando se presuponga F) es un objeto $X \in \mathbb{C}$ junto con un morfismo $c : X \to FX$. Generalmente diremos que X es el espacio de estados y c la estructura coalgebraica o de transición.*

2. *Diremos que $f : X \to Y$ es un homomorfismo entre las coálgebras $c : X \to TX$ y $d : Y \to TY$ si el siguiente diagrama es conmutativo:*

$$\begin{array}{ccc} TX & \xrightarrow{Tf} & TY \\ {\scriptstyle c}\uparrow & & \uparrow{\scriptstyle d} \\ X & \xrightarrow[f]{} & Y \end{array}$$

Nota 2.2.1. *Cabe destacar que en esta tesis únicamente hemos usado las denominadas* coálgebras funtoriales*, en contraposición con las* coálgebras de Eilenberg-Moore *para comonoides [Mac98].*

Las propias F-coálgebras, junto con los homomorfismos entre ellas, forman la categoría $\mathbf{CoAlg}(F)$. En concreto, una *coálgebra final* es un objeto final de la categoría $\mathbf{CoAlg}(F)$, es decir, una coálgebra $\gamma : Z \to FZ$ tal que para cualquier otra coálgebra $c : X \to FX$ existe un único homomorfismo $\mathsf{beh}_c : X \to Z$ tal que el siguiente diagrama es conmutativo:

$$\begin{array}{ccc} FX & \xrightarrow{F(\mathsf{beh}_c)} & FZ \\ {\scriptstyle c}\uparrow & & \uparrow{\scriptstyle \gamma} \\ X & \xrightarrow[\mathsf{beh}_c]{} & Z \end{array}$$

El siguiente lema fue enunciado y demostrado por Lambek [Lam68] y es de vital importancia a la hora de tratar con objetos terminales (ya sean iniciales o finales).

Lema 2.2.1. *Dada un álgebra final $\gamma : FZ \to Z$ para un endofuntor F, la estructura $\gamma : Z \to FZ$ es un isomorfismo, es decir, se tiene que FZ es isomorfo a Z.*

Nota 2.2.2. *El lema 2.2.1 es también aplicable a las coálgebras finales ya que el concepto de isomorfismo es* auto-dual*, es decir, la definición dual de un isomorfismo coincide con la definición original de isomorfismo. Por tanto las coálgebras finales también son isomorfismos.*

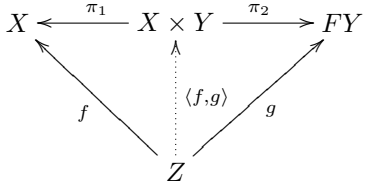
El homomorfismo $\mathsf{beh}_c(x)$ es especialmente interesante pues indica el comportamiento del estado x y, como veremos más adelante en la sección 2.2.2, si $\mathsf{beh}_c(x) = \mathsf{beh}_c(y)$ los estados x e y serán bisimilares.

Existen una serie de operaciones funtoriales básicas que son los productos, coproductos, la exponenciación y el conjunto potencia, que aunque no pueden construirse en todas las categorías (sí que pueden construirse en **Sets**), en el caso de las tres primeras, siempre tienen la propiedad de ser únicas salvo isomorfismo.

Definición 2.2.6 (Producto)**.** *En una categoría $\mathbb{C}$, el producto de los objetos $X, Y \in \mathbb{C}$ es un objeto $X \times Y \in \mathbb{C}$ con dos morfismos llamados proyecciones*

$$X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$$

tales que son universales, es decir, que para todo par de flechas $f : Z \to X$ y $g : Z \to Y$ en $\mathbb{C}$ existe un único morfismo $\langle f, g\rangle : Z \to X \times Y$ que hace que el siguiente diagrama sea conmutativo:



Análogamente, dados dos morfismos $f : X \to X'$ y $g : Y \to Y'$, y las proyecciones $\pi_1 : X \times Y \to X$ y $\pi_2 : X \times Y \to Y$, se define el producto de los morfismos $f \times g : X \times Y \to X' \times Y'$ como

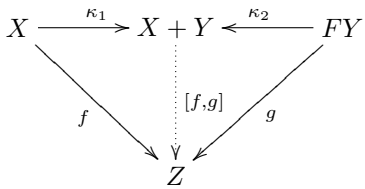
$$f \times g = \langle f \circ \pi_1, g \circ \pi_2 \rangle .$$

El producto $f \times g$ está caracterizado por las dos siguientes propiedades: $\pi_1 \circ (f \times g) = f \circ \pi_1$ y $\pi_2 \circ (f \times g) = g \circ \pi_2$.

Definición 2.2.7 (Coproducto)**.** *En una categoría $\mathbb{C}$, el coproducto de $X, Y \in \mathbb{C}$ es un objeto $X + Y \in \mathbb{C}$ con dos morfismos llamados inyecciones*

$$X \xrightarrow{\kappa_1} X + Y \xleftarrow{\kappa_2} Y$$

que satisfacen la propiedad universal, es decir, para todo par de flechas $f : X \to Z$ y $g : Y \to Z$ en $\mathbb{C}$ existe un único morfismo $[f, g] : X + Y \to Z$ que hace que el siguiente diagrama sea conmutativo



Análogamente, dados dos morfismos $f : X \to X'$ y $g : Y \to Y'$, y las inyecciones $\kappa_1 : X \to X + Y$ y $\kappa_2 : Y \to X + Y$, se define la suma de los morfismos $f + g : X + Y \to X' + Y'$ como

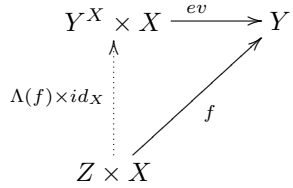
$$f + g = [\kappa_1 \circ f, \kappa_2 \circ g] .$$

La suma $f + g$ está caracterizado por las dos siguientes propiedades: $\kappa_1 \circ f = (f + g) \circ \kappa_1$ y $\kappa_2 \circ g = (f + g) \circ \kappa_2$.

Definición 2.2.8 (Exponenciación)**.** *En una categoría $\mathbb{C}$ con producto $\times$, la exponenciación de los objetos $X, Y \in \mathbb{C}$ es un objeto $Y^X \in \mathbb{C}$ con un morfismo de evaluación*

$$Y^X \times X \stackrel{ev}{\to} Y$$

que, de nuevo, satisface la propiedad universal. Es decir, se tiene que para toda flecha $f : Z \times X \to Y$ en $\mathbb{C}$, existe un único morfismo $\Lambda(f) : Z \to Y^X$ que hace que el siguiente diagrama sea conmutativo



Análogamente, dado un morfismo $h : X \to Y$ y un objeto fijo A, se define la exponenciación $h^A : X^A \to Y^A$ como el morfismo

$$h^A = \Lambda(h \circ ev)\,.$$

Es decir, h^A es tomar el morfismo Λ sobre $h \circ ev : X^A \times A \stackrel{ev}{\to} X \stackrel{h}{\to} Y$. Además, la exponenciación h^A está caracterizada por la propiedad: $h \circ ev = ev \circ (h^A \times id_A)$.

Existen varias maneras de definir el conjunto potencia, incluso en la categoría **Sets**. Nosotros hemos usado la siguiente definición: dado un conjunto X se define el *conjunto potencia de X*, denotado por $\mathcal{P}(X)$, de la siguiente manera $\mathcal{P}(X) = \{U \mid U \subseteq X\}$, es decir, como el conjunto de todos los subconjuntos de X. Si tenemos una función $f : X \to Y$ entonces existe otra función $\mathcal{P}(f) : \mathcal{P}(X) \to \mathcal{P}(Y)$ tal que para $U \subseteq X$

$$\begin{array}{rcl} \mathcal{P}(f)(U) = f(U) & = & \{f(x) \mid x \in U\} \\ & = & \{y \in Y \mid \exists x \in X.\ f(x) = y \wedge x \in U\}\,. \end{array}$$

De esta forma hemos definido un funtor $\mathcal{P}(\cdot) : \mathbf{Sets} \to \mathbf{Sets}$. También denotaremos a $f(U)$, la imagen de U bajo f, como $\coprod_f U$ usando la notación de las categorías fibradas [Gro57, Jac99].

Si bien los posibles funtores conjunto potencia en **Sets** actúan siempre igual sobre los conjuntos, tenemos distintas opciones al definirlos sobre las funciones. Por ejemplo, podríamos haberlo considerado como un *funtor contravariante*, es decir $\mathcal{P} : \mathbf{Sets}^{op} \to \mathbf{Sets}$, sin más que considerar la imagen inversa de la función: dada $f : X \to Y$ consideramos $\mathcal{P}(f) : \mathcal{P}(Y) \to \mathcal{P}(X)$ y para un $U \subseteq Y$

$$\mathcal{P}(f)(U) = \{x \mid f(x) \in U\}\,.$$

Ya estamos en disposición de mostrar una serie de ejemplos de estructuras que pueden definirse sencillamente como coálgebras.

Ejemplo 2.2.3 En primer lugar, ya vimos al principio de la sección tres ejemplos sencillos de coálgebras: la máquina que tenía un único botón y que podía romperse tras presionarlo

un número indeterminado de veces; la máquina de dos botones que modelizaba las listas posiblemente infinitas de elementos de un conjunto A; y la clase `Point`. La primera está definida por el functor $F = 1 + id$, la segunda por $F' = 1 + (A \times id)$, mientras que la tercera tiene como functor $F'' = \mathbb{R} \times \mathbb{R} \times id^{\mathbb{R}\times\mathbb{R}}$.

También hemos visto cómo se definen los sistemas de transiciones como coálgebras, ya que eso fue precisamente lo que Aczel modelizó. Es decir, un LTS no es más que una coálgebra para el funtor $F = \mathcal{P}(A \times id)$ o, equivalentemente, para el funtor $\mathcal{P}(id)^A$. Así diremos que $c : X \to \mathcal{P}(X)^A$ es un LTS y, por comodidad, pese a que estemos en el caso coalgebraico, seguiremos usando la notación $x \xrightarrow{a} x'$ para indicar que $x' \in c(x)(a)$.

Un caso más general de LTS son los *autómatas no-deterministas*, que tienen una función de transición $\delta : S \to \mathcal{P}(S)^A$ y una función de salida $\epsilon : S \to B$, donde A es el alfabeto de entrada y B el de salida. Los autómatas no-deterministas son coálgebras definidas como el producto de las dos funciones anteriores: $Nd : S \to \mathcal{P}(S)^A \times B$.

Así, una estructura de Kripke no es más que un caso particular de autómata no-determinista en el que la función de salida es $\epsilon : S \to \mathcal{P}(AP)$, donde AP es un conjunto de proposiciones atómicas.

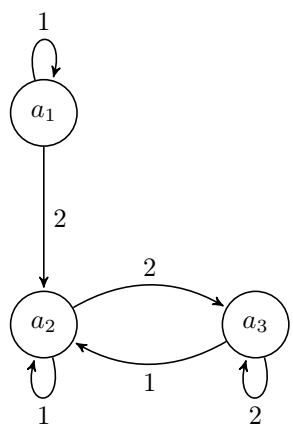
También es posible definir los árboles binarios (infinitos) con nodos en el conjunto A como una coálgebra $t : X \to A \times X \times X$.

Si en vez de t tomásemos el funtor $t' = 1 + (A \times id \times id)$ se podrían construir tanto árboles binarios infinitos como finitos.

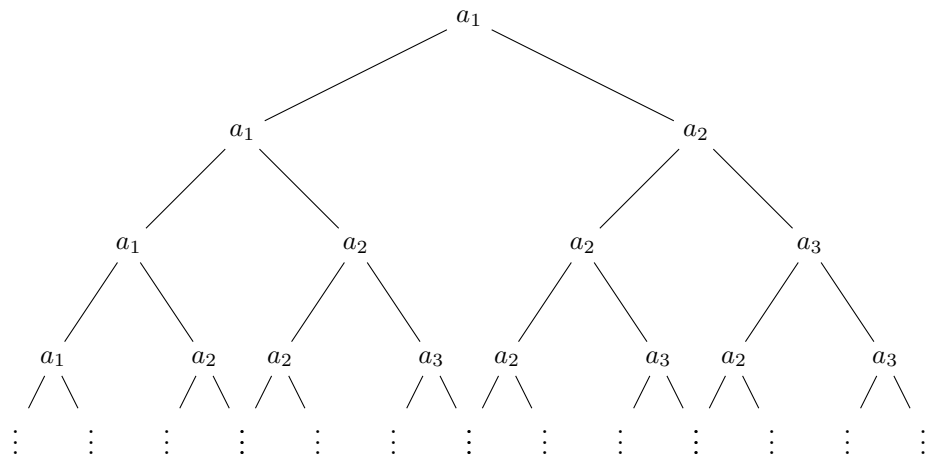
En concreto, si tomamos $X = \{x_1, x_2, x_3\}$ y $A = \{a_1, a_2, a_3\}$, podemos definir la coálgebra $c : X \to A \times X \times X$ con

$$\begin{aligned} c(x_1) &= (a_1, x_1, x_2) \\ c(x_2) &= (a_2, x_2, x_3) \\ c(x_3) &= (a_3, x_2, x_3) \end{aligned}$$

Es decir,



donde hemos usado un índice para diferenciar el hijo izquierdo (el 1) del hijo derecho (el 2). Como se observa, la representación gráfica de la coálgebra c es un grafo. De hecho, del grafo obtenemos tres árboles binarios infinitos según empecemos a desarrollar las computaciones en x_1, x_2 o x_3. En particular, empezando en x_1 obtenemos:



Todos los ejemplos de funtores anteriores corresponden a una subclase especial de estos. Se trata de la subclase de *funtores polinomiales* (a veces llamados también *funtores polinomiales de Kripke*, y abreviados como KPF), que está formada por todos aquellos que se pueden construir a partir de los funtores identidad, constante, producto, coproducto, exponenciación para un conjunto constante, y el funtor conjunto potencia. La utilidad de los KPF es que permiten definir una gran variedad de estructuras y, además, debido a su definición estructurada permiten demostraciones por inducción estructural [Jac].

Una variante muy usada en la literatura es la subclase finKPF, cuya única diferencia con la clase KPF es que se define utilizando las partes finitas, $\mathcal{P}_\omega$, en vez de $\mathcal{P}$. La razón para ello es que, como el mismo Aczel demostraba, no existe una coálgebra final para el funtor $\mathcal{P}$. Esto es debido a que, como destacábamos en la nota 2.2.2, las coálgebras finales son isomorfismos, con lo que si existiese tal coálgebra final $\gamma : Z \to \mathcal{P}Z$ para este funtor $\mathcal{P}$, $\mathcal{P}Z$ tendría que tener el mismo cardinal que Z, llegándose así a una contradicción.

2.2.2. Bisimulación entre coálgebras

En esta sección vamos a tratar y relacionar las distintas definiciones de bisimulación que se han ido proponiendo a lo largo de los años, así como su relación con la coinducción.

Bisimulación como coinducción

Intuitivamente, un conjunto X está definido *inductivamente* si es la menor solución de una inecuación de un cierto tipo. El *principio de inducción* dice que cualquier otro conjunto que sea solución de esa inecuación contiene al conjunto X. Este principio de inducción lleva siendo usado por los matemáticos desde los tiempos de Euclides (valga como ejemplo la demostración de que el conjunto de los números primos es infinito) y coincide con la inicialidad de las álgebras: la existencia del homomorfismo *es* la definición inductiva, mientras que la unicidad *es* el principio de prueba por inducción.

Si dualizamos la noción anterior, obtenemos que X está definido *coinductivamente* si es la mayor solución de una inecuación de un cierto tipo, mientras que el *principio*

de coinducción dice que cualquier otro conjunto que sea solución de esa inecuación está contenido en el conjunto X. Análogamente, el principio de coinducción coincide con la finalidad de las coálgebras: la existencia del homomorfismo *es* la definición coinductiva, mientras que la unicidad *es* el principio de prueba por coinducción. Además, ya hemos comentado en varias ocasiones que las pruebas por bisimulación son una instancia del principio de prueba mediante coinducción, pero ¿cómo se relacionan exactamente la bisimulación y la coinducción?

En primer lugar, debemos tener en cuenta que las coálgebras únicamente nos permiten observar su comportamiento o avanzar a otro estado. Como decíamos tras la definición de coálgebra en la sección 2.2.1, el (único) homomorfismo de una coálgebra hacia la coálgebra final beh_c nos informa del comportamiento de cada estado. Debido a que la coálgebra final es "la más pequeña", en ella no hay estados repetidos, es decir, no pueden existir dos estados distintos z_1 y z_2 que tengan el mismo comportamiento. En conclusión, si en una coálgebra hay dos estados x e y cuyos comportamientos observacionales son idénticos (es decir, son dos estados *bisimilares*), entonces $\mathsf{beh}_c(x) = \mathsf{beh}_c(y)$. Este resultado es el *principio de prueba mediante coinducción* y fue estudiado por Rutten y Turi en [RT92].

Este principio de prueba por coinducción puede reformularse de otras muchas maneras. Valga como ejemplo la que se da en [JR97]:

> Dada una coálgebra final $\gamma : Z \to FZ$, para todo par de elementos $z, z' \in Z$ se tiene que si zRz' para una bisimulación R, entonces $z = z'$.

Consideremos el funtor $\mathcal{P}_\omega(A \times id)$ para los LTS finitamente ramificados, y su coálgebra final $\gamma : Z \to \mathcal{P}_\omega(Z)$. Siendo $R \subseteq Z \times Z$ una bisimulación, vamos a dotarla de una estructura coalgebraica sin más que tomar la coálgebra $b : R \to \mathcal{P}_\omega(A \times R)$, definida para los $r = (p, q) \in R$ como:

$$b(r) = \{\langle a, (p', q')\rangle \mid p \xrightarrow{a} p' \text{ y } q \xrightarrow{a} q'\}\,.$$

Ahora, las proyecciones $\pi_1 : R \to Z$ y $\pi_2 : R \to Z$ son homomorfismos entre coálgebras, por lo que deben coincidir debido a la finalidad de γ. Hemos demostrado así que si tenemos pRq para una bisimulación en Z, entonces $p = q$, es decir, el principio de coinducción. De este modo se observa que la bisimulación de Park [Par81] es una instancia del método de coinducción.

Bisimulación como *spans*

El camino que tomó Peter Aczel en su libro "Non-well-founded Sets" [Acz88], pese a que no llegase a dar una definición coalgebraica explícita de bisimulación hasta [AM89], fue el que hemos descrito en el punto anterior. Aczel se inspiró en el famoso trabajo de Milner de 1982 [Mil82] observando en él que agentes diferentes podían tener el mismo comportamiento (expresado por un sistema de transiciones etiquetadas). Se dio cuenta así de que al considerar los sistemas de transiciones que representaban el comportamiento de los agentes y hacer el cociente con la relación de *bisimulación*, se obtendrían otros sistemas de transiciones que podían ser vistos como la interpretación matemática del lenguaje de los agentes definido en [Mil82].

Considerando los sistemas de transiciones etiquetados como coálgebras para el funtor $F = \mathcal{P}_\omega(A \times id)$, Aczel se percató de que al hacer ese cociente entre la coálgebra y la relación de bisimulación, la coálgebra que se obtenía era la coálgebra final para F. De este modo, Aczel propuso el *teorema de la coálgebra final* [Acz88], que dice que para todo funtor *set-based*[3] existe una coálgebra final.

Como ya hemos comentado anteriormente, Aczel no llegó a probar el *teorema de la coálgebra final* hasta un año después en [AM89], junto con Nax Mendler. En aquel artículo, ambos autores cerraban además la conexión con el trabajo de Milner y la bisimulación al proponer una definición de esta última puramente coalgebraica. Esta definición hacía uso de los *spans* que, intuitivamente, no son más que las generalizaciones de las relaciones $(Y \leftarrow R \rightarrow X)$, como queda patente en la siguiente definición extraída de [AM89].

Definición 2.2.9 (Bisimulación Aczel-Mendler)**.** *Sean $c : X \to FX$ y $d : Y \to FY$ dos coálgebras y $R \subseteq X \times Y$ una relación. Decimos que R es una bisimulación si existe una coálgebra $e : R \to FR$ de manera que las proyecciones $r_1 : R \hookrightarrow X$ y $r_2 : R \hookrightarrow Y$ son homomorfismos, es decir, hacen que el siguiente diagrama sea conmutativo:*

$$\begin{array}{ccccc} X & \xleftarrow{r_1} & R & \xrightarrow{r_2} & Y \\ \downarrow{\scriptstyle c} & & \downarrow{\scriptstyle e} & & \downarrow{\scriptstyle d} \\ FX & \xleftarrow{Fr_1} & FR & \xrightarrow{Fr_2} & FY \end{array}$$

Bisimulación como relación

Al margen de la caracterización por medio de la igualdad en la coálgebra inicial, y la anterior facilitada por Aczel y Mendler, existe otra definición coalgebraica de bisimulación, propuesta por Bart Jacobs [Jac]. La misma usa la técnica de *alzamiento de relaciones* [Her93, HJ98]. La idea para modelizar la bisimulación sigue siendo esencialmente la clásica: una relación R es una bisimulación si cuando un par (x, y) pertenece a él, entonces sus sucesores, dados por $c(x)$ y $d(y)$ (siendo c y d dos coálgebras), siguen perteneciendo a la relación. No obstante, x y $c(x)$ (y análogamente y y $d(y)$) no se mueven en el mismo dominio, con lo que se deberá definir un operador que "eleve" la relación $R \subseteq X \times Y$ a una relación $R' \subseteq FX \times FY$.

Definición 2.2.10 (Alzamiento)**.** *Dados un funtor F, conjuntos X e Y, y una relación $R \subseteq X \times Y$ cuyas proyecciones son $\langle r_1, r_2 \rangle : R \hookrightarrow X \times Y$, se define* $\mathrm{Rel}(F)(R)$ *como la imagen del par*

$$\langle F(r_1), F(r_2) \rangle : FR \to FX \times FY\,.$$

Para el caso de los funtores polinomiales existe una definición estructural del alzamiento de relaciones [Jac], que simplifica notablemente su uso. Por ejemplo, para el caso del funtor $F = \mathcal{P}(id)^A$, que caracteriza los LTS, obtenemos la siguiente particularización:

[3] Aquí estamos citando el teorema más general en [AM89], donde se trabaja con la categoría de las **Clases** (y no con **Sets**): los objetos son clases y los morfismos funciones entre clases. En **Clases** un funtor es *set-based* si para cada clase A y cada elemento $a \in FA$ existe un conjunto $A_0 \subseteq A$ y $a_0 \in A_0$ tal que $a = F(\iota a_0)$, donde ι es la inclusión de A_0 en A.

$$\mathrm{Rel}(\mathcal{P}(id)^A)(R) \quad = \{(f,g) \mid \forall a \in A.\, (f(a), g(a)) \in \{(U,V) \mid \quad \forall u \in U.\, \exists v \in V.\, uRv \wedge \\ \forall v \in V.\, \exists u \in U.\, uRv\}\}\,.$$

Ahora, las bisimulaciones, según [Jac], se definen de la forma siguiente:

Definición 2.2.11. *Sean $c : X \to FX$ y $d : Y \to FY$ dos coálgebras y $R \subseteq X \times Y$ una relación. Diremos que R es una bisimulación si*

$$(x,y) \in R \Longrightarrow (c(x), d(y)) \in \mathrm{Rel}(F)(R),$$

es decir, si

$$R \subseteq (c,d)^{-1}\mathrm{Rel}(F)(R)\,.$$

Nota 2.2.3. *Como se indica en [HJ04], las bisimulaciones pueden verse también como* $\mathrm{Rel}(F)$*-coálgebras en la categoría* **Rel** *de relaciones. Es decir, una bisimulación es un morfismo en* **Rel** *sobre dos morfismos en* **Sets***, que coinciden con las dos coálgebras $c : X \to FX$ y $d : Y \to FY$.*

Es especialmente importante observar que esta definición de bisimulación es, en cierto sentido, una definición local. Si se observa con cuidado la propiedad que debe cumplir una relación para ser bisimulación, observamos dicha localidad, en el sentido de que la condición involucra únicamente a un par (x, y) y sus sucesores inmediatos, sin que sea posible establecer relaciones entre términos no ligados por la "cercanía en X" que expresan las coálgebras c y d. En contraposición, la definición propuesta por Aczel y Mendler tiene un carácter más global, pues la bisimulación queda caracterizada por una coálgebra "entera".

También se puede decir (como el propio Jacobs recoge en [Jac]) que la definición 2.2.11 es de un corte lógico, y que caracteriza las bisimulaciones como una relación que cumple cierta propiedad especial, de un modo parecido a cómo la definición clásica también considera a la bisimulación como una relación "especial". Por ello, se puede justificar que esta es una definición más natural que la definición 2.2.9, que echa mano de los *spans* y define la bisimulación como una estructura (coálgebra) especial. Independientemente de las justificaciones y de las sutiles diferencias entre las dos definiciones, como se demuestra en [Jac, teorema 3.3.2] ambas definiciones resultan ser equivalentes entre sí, por lo que no tiene sentido litigar sobre "cuál es la más correcta".

Algunos ejemplos de bisimulaciones

Una de las grandes ventajas de las tres definiciones categóricas de bisimulación que hemos visto, es que nos permiten definir el concepto de bisimulación no solo para los LTS, sino para una amplísima variedad de estructuras matemáticas. Pero, puesto que con la definición de Jacobs para el caso de los KPF la noción de bisimulación puede hacerse explícita, vamos a aprovecharnos de ello presentando detalladamente algunos ejemplos haciendo uso de la definición 2.2.11.

Ejemplo 2.2.4

1. Vamos a considerar autómatas deterministas, es decir, coálgebras para el funtor $F = id^A \times \{0,1\}$, y las palabras de A^* que aceptan. Diremos que una palabra $\alpha = a_1 \dots a_k$ es aceptada por un estado x de un autómata si a partir de x, realizando las acciones que define α, se llega a un estado de aceptación. Expresado formalmente, si $c(x) = (\delta(x), \epsilon(x))$ entonces α es una palabra aceptada si tomando $x_1 = \delta(x)(a_1)$, $x_2 = \delta(x_1)(a_2)$, hasta llegar a $x_k = \delta(x_{k-1})(a_k)$ con $\epsilon(x_k) = 1$.

   Intuitivamente, si $c = \langle \delta, \epsilon \rangle : X \to X^A \times B$ y $d = \langle \delta', \epsilon' \rangle : Y \to Y^A \times B$ son dos autómatas deterministas con $B = \{0,1\}$ y existe una bisimulación R tal que xRy, entonces los lenguajes de aceptación a partir de x y de y deben coincidir. En efecto, como xRy, por definición se tiene que $c(x)\,\mathrm{Rel}(id^A \times \{0,1\})(R)\,d(y)$, donde

$$\mathrm{Rel}(id^A \times \{0,1\})(R) = \{((u_1,u_2),(v_1,v_2)) \mid u_2 = v_2 \wedge \forall a.\, u_1(a)\ R\ v_1(a)\},.$$

   Esto quiere decir que $\epsilon(x) = \epsilon'(y)$ (es decir, o ambos estados son de rechazo o ambos de aceptación) y para todo $a \in A$ debe tenerse que si $x \xrightarrow{a} x'$ y también $y \xrightarrow{a} y'$, entonces $x'Ry'$. Con lo que si desde x se acepta una palabra, entonces desde y también se aceptará, y viceversa (ya que se deben reproducir las mismas acciones en ambos lados).

2. Veamos un ejemplo de bisimulación entre árboles binarios finitos e infinitos. Consideramos entonces coálgebras para el funtor $F = 1 + (A \times id \times id)$. En este caso la bisimulación es esencialmente la igualdad, pues si R es una bisimulación tal que xRy, entonces $c(x)\mathrm{Rel}(F)(R)d(y)$ significa que, o bien $c(x) = * = d(y)$, siendo $*$ el único elemento de 1; o, si no tendremos $c(x) = (a, x, x')$ y $d(y) = (b, y, y')$, siendo $a = b$, xRy y $x'Ry'$.

Aunque fuera del formalismo de Jacobs, las bisimulaciones probabilísticas de Larsen y Skou que vimos en la sección 2.1.2 tienen también su generalización al mundo categórico. Esta generalización la llevaron a cabo Vink y Rutten en 1999 [dVR99]. En esa publicación los autores extienden el conjunto $\mathcal{D}(X)$ de distribuciones probabilísticas sobre X a las funciones (para así definir un funtor), del siguiente modo: dada $f : S \to T$ y una distribución probabilística μ en S, entonces $\mathcal{D}(f) : \mathcal{D}(S) \to \mathcal{D}(T)$ está definida como $\mathcal{D}(f)(\mu)(t) = \sum_{f(s)=t} \mu(s)$.

Por lo tanto, un sistema de transiciones probabilístico es una coálgebra para el funtor $F = (\mathcal{D}(id) + 1)^{Act}$, aunque, por claridad, los autores acaban eliminando la posibilidad de terminación y las etiquetas de las acciones, trabajando únicamente sobre el funtor $\mathcal{D}(id)$. Así, dadas dos coálgebras $\alpha : S \to \mathcal{D}(S)$ y $\beta : T \to \mathcal{D}(T)$, Rutten y Vink definen una bisimulación probabilística desde el punto de vista coalgebraico como una relación, $R \subseteq S \times T$, tal que si sRt entonces existen U y V tales que $\sum_{u \in U} \alpha(s)(u) = \sum_{v \in V} \beta(t)(v)$, con $\pi_1^{-1}(U) = \pi_2^{-1}(V)$, donde π_1 y π_2 son las proyecciones de R.

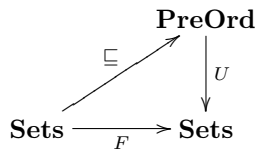
Los autores demuestran que esta noción de bisimulación probabilística coincide con la clásica, y con la noción de bisimulación de Aczel y Mendler para el funtor $\mathcal{D}$.

2.2.3. Simulaciones

Como veremos a lo largo de esta sección, existen diversos modos de definir las simulaciones entre coálgebras. Empezaremos recogiendo el método ideado por Hughes y Jacobs [HJ04], que es el que hemos ido usando a lo largo de nuestra investigación.

En primer lugar, para definir las simulaciones desde el punto de vista coalgebraico tendremos que introducir previamente el concepto de *orden asociado a un funtor* [HJ04].

Definición 2.2.12 (Orden asociado a un funtor)**.** *Si $F : \mathbf{Sets} \to \mathbf{Sets}$ es un endofuntor, un orden de F es otro funtor, $\sqsubseteq : \mathbf{Sets} \to \mathbf{PreOrd}$, tal que el siguiente diagrama es conmutativo*



donde U es el funtor olvidadizo, es decir, en este caso, el funtor que "olvida" la estructura del preorden para quedarse únicamente con su conjunto subyacente.

Conviene destacar que el orden $\sqsubseteq$ asociado al funtor F asocia a cada conjunto X un preorden $\sqsubseteq_X \subseteq FX \times FX$, que debe preservarse por renombramiento, es decir, que para toda función entre conjuntos $f : X \to Y$, la función $F(f) : FX \to FY$ es monótona con respecto a $\sqsubseteq_X$ y $\sqsubseteq_Y$. Así si $u \sqsubseteq_X u'$ se debe tener $Ff(u) \sqsubseteq_Y Ff(u')$ para toda f. Intuitivamente esto significa que el orden no puede usar la información de los estados X, sino solamente usar la estructura del funtor F.

Rápidamente se observa que un posible orden es la igualdad, es decir, $u \sqsubseteq v$ si y solo si $u = v$, o también el orden trivial que considera que todo par de elementos están relacionados entre sí. Pero podemos considerar otros órdenes no triviales como, por ejemplo, la inclusión punto a punto para el funtor $F = \mathcal{P}(id)^A$, en la que se tiene $u \sqsubseteq v$ si y solo si para todo $a \in A$ tenemos $u(a) \subseteq v(a)$.

Apoyándose en este orden y en el alzamiento de relaciones, Hughes y Jacobs definen las simulaciones desde el punto de vista coalgebraico.

Definición 2.2.13 (Simulación coalgebraica)**.** *Sea* $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ *definida por*

$$\sqsubseteq_Y \circ \mathcal{R}el(F)(R) \circ \sqsubseteq_X \ .$$

Diremos que $R \subseteq X \times Y$ es una simulación entre las coálgebras $X \xrightarrow{c} FX$ y $Y \xrightarrow{d} FY$, si

$$\begin{array}{rcl} (x,y) \in R & \Longrightarrow & (c(x), d(y)) \in \mathcal{R}el_{\sqsubseteq}(F)(R) \\ & \Longleftrightarrow & \exists u, v.\ (c(x) \sqsubseteq_X u) \wedge (v \sqsubseteq_Y d(y)) \wedge (u,v) \in \mathcal{R}el(F)(R) \end{array}$$

o, equivalentemente,

$$R \subseteq (c \times d)^{-1}(\mathcal{R}el_{\sqsubseteq}(F)(R)).$$

Nota 2.2.4. *Existe una* inmersión completa y plena *(en inglés* a full and faithful embedding*) entre* **PreOrd** $\hookrightarrow$ **Rel** *[JH03], es decir, se puede ver a la categoría* **PreOrd** *como una subcategoría de relaciones* **Rel**. *Por tanto, como en el caso de las bisimulaciones, las simulaciones pueden ser vistas como* $\mathrm{Rel}_{\sqsubseteq}(F)$*-coálgebras en la categoría* **Rel**.

Quizá lo más sorprendente de esta definición es que desde el punto de vista coalgebraico la bisimulación es un caso particular de la simulación, puesto que basta considerar para ello el orden identidad. También observamos que esta definición permite que se definan simulaciones un tanto extrañas. Por ejemplo, si consideramos los árboles binarios definidos como coálgebras del funtor $F = 1 + (A \times id \times id)$ y definimos el orden $\sqsubseteq_X \subseteq (1 + (A \times X \times X)) \times (1 + (A \times X \times X))$ como

$$u \sqsubseteq_X v \text{ si y solo si } v = * \text{ o } u = v\,,$$

obtenemos que una simulación entre árboles considera que un árbol infinito es simulado por otro finito. Evidentemente, para soslayar en este caso la dificultad a nivel intuitivo que tal hecho supone, basta con considerar el orden opuesto. Como consecuencia tenemos una de las primeras sorpresas que nos da la definición de Hughes y Jacobs ya que obtenemos que la simulación no induce una dirección "natural" de "simulación", sino que ambas direcciones son igualmente correctas, según tengamos en cuenta un orden $\sqsubseteq$ o su inverso $\sqsupseteq$.

Por ejemplo, las simulaciones entre sistemas de transiciones son las simulaciones coalgebraicas para el orden $\sqsubseteq_X \subseteq \mathcal{P}^A(X) \times \mathcal{P}^A(X)$ definido por

$$u \sqsubseteq_X v \text{ si y solo si } u(a) \subseteq v(a)\,.$$

Con este orden se recoge también la idea de que un LTS p simula a otro LTS q si p puede hacer lo que hace q y, posiblemente, más cosas. Eso sí, en este mundo coalgebraico, la filosofía es radicalmente distinta a la clásica, como se pone de manifiesto al observar detalladamente la condición para que R sea una simulación entre los estados x e y de dos sistemas de transiciones no etiquetados,

$$q(x) \subseteq_X u \,\mathrm{Rel}(\mathcal{P}id)(R)\, v \subseteq_Y p(y)\,. \tag{2.1}$$

$p(y)$ (respectivamente, $q(x)$) es el conjunto de sucesores del estado y (respectivamente, de x). El uso del orden $\subseteq$ en la ecuación 2.1 nos indica que, podemos o bien mayorar el conjunto de sucesores de x tomando otro más grande u, o minorar el conjunto de sucesores de y por medio de v; o ambas cosas simultáneamente, para luego comprobar que u y v son bisimilares. Es decir, p simula a q siempre que, o bien p pueda desprenderse de aquellas transiciones "extra" que le impiden ser bisimilar a q; o siempre que q pueda "añadirse" las transiciones extra que necesita para ser bisimilar a p.

Observamos ahora que siempre se podrán eliminar transiciones de un LTS, mientras que en general no siempre se podrán añadir las que pudiéramos necesitar, pues al efecto estamos a expensas de que el conjunto X contenga los elementos adecuados para mayorar $q(x)$ vía el u que necesitemos. En segundo lugar, la definición de simulación sigue manteniendo la localidad de la que hacía gala la bisimulación: el orden nos permite "eliminar" transiciones en cada paso, pero es más importante observar que esa eliminación no se

hace sobre la coálgebra, sino solo "de manera local" sobre los conjuntos u y v. Es decir, si durante el juego de la simulación debemos volver a comprobar el estado x, este "seguirá teniendo" como sucesores los elementos de $q(x)$, y nunca los elementos adicionales que habíamos introducido en algún otro paso de la aplicación de la definición.

Pese a que *a priori* cualquier orden categórico $\sqsubseteq$ es válido para ser usado en la definición 2.2.13, no todos los órdenes generan buenas propiedades en el alzamiento $\mathrm{Rel}_{\sqsubseteq}(F)$. En concreto, nos encontramos con la desagradable sorpresa de que es posible que la composición de simulaciones no sea siempre una simulación, en cuyo caso la relación de similitud definida por ese orden podría no ser transitiva, por lo que no estaríamos definiendo una relación de similitud "razonable". En [HJ04] se incluye un ejemplo no demasiado complejo de simulación coalgebraica no transitiva. Se trata de la *simulación lexicográfica* definida para el funtor $FX = 2 \times X$ de la siguiente manera: sean $u = (n, x)$ y $v = (m, y)$ en FX, el orden lexicográfico $\sqsubseteq_{lex}$ viene dado por

$$u \sqsubseteq_{lex} v \iff n < m \text{ o } (n = m \text{ y } x = y).$$

De hecho no es adecuado llamar a esa relación ni *simulación* (ya que no es una relación transitiva) ni *lexicográfica* porque, de ser lexicográfica de verdad, entonces esperaríamos que toda lista l que empiece por 1 simula (es mayor) que cualquier lista t que empiece por 0, como se muestra en la figura 2.7, pero esto claramente no es cierto para las $\sqsubseteq_{lex}$-simulaciones.

Figura 2.7: La lista t es menor que la lista l con el orden lexicográfico $\preceq_{lexicographic}$, pero no existe ninguna $\sqsubseteq_{lex}$-simulación que relacione x e y.

¿Cómo podemos entonces garantizar que una similitud es "razonable"? La solución consiste en restringir los órdenes posibles. Esta restricción la realizaron Hughes y Jacobs en varias etapas, que pasamos a detallar a continuación.

Originariamente, en 2003 [JH03] para poder garantizar que la composición de simulaciones sea siempre una simulación se recurría a una condición suficiente, basada en que el orden $\sqsubseteq$ preserve la composición del alzamiento (en inglés que el orden sea *composition-preserving*):

Definición 2.2.14 ([JH03] 5.2)**.** *Un funtor F con orden $\sqsubseteq$* preserva la composición *para el alzamiento $\mathrm{Rel}_{\sqsubseteq}(F)$ siempre que se tenga la siguiente condición:*

$$\mathrm{Rel}_{\sqsubseteq}(F)(R \circ S) = \mathrm{Rel}_{\sqsubseteq}(F)(R) \circ \mathrm{Rel}_{\sqsubseteq}(F)(S)\,.$$

Aunque es cierto que la condición de preservar composiciones es bastante natural, en la versión revisada del artículo anterior [HJ04] se definió otra condición algo menos intuitiva pero más general, que garantizaba que el alzamiento de relaciones $\mathrm{Rel}_{\sqsubseteq}(F)$ preservara composiciones. Esa condición se llamó *estabilidad* (*stability* en inglés):

Definición 2.2.15 ([HJ04] 4.3)**.** *Se dice que un funtor con orden $\sqsubseteq$ es* estable *si el alzamiento de relaciones $\mathrm{Rel}_{\sqsubseteq}(F)$ conmuta con la sustitución, es decir, si para todas $f : X \to Z$ y $g : Y \to W$, se tiene*[4]*:*

$$\mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R)) = (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R)) .$$

Esta condición de *estabilidad* tiene dos importantes problemas. El primero es que es una condición puramente *ad hoc*, sacada como una (casi trivial) generalización de lo que se necesita demostrar para que el alzamiento $\mathrm{Rel}_{\sqsubseteq}(F)$ preserve composiciones, en donde basta considerar como relación R la igualdad, y como f y g sus proyecciones. Además, la condición de *estabilidad* resulta, en general, complicada de probar. Esto llevó a los autores a considerar otra condición más fuerte que implicase la *estabilidad*. Esta condición no recibió nombre alguno en [HJ04], pero nosotros la bautizamos como *estable por la derecha* (o *right-stability* en inglés) en la segunda publicación del capítulo 5:

Definición 2.2.16 ([HJ04])**.** *Decimos que un functor F con orden $\sqsubseteq$ es* estable por la derecha *si, para toda función $f : X \to Y$, tenemos*[5]

$$(id \times Ff)^{-1} \sqsubseteq_Y \ \subseteq \coprod_{Ff \times id} \sqsubseteq_X . \tag{2.2}$$

donde, $\coprod_{Ff\times id} \sqsubseteq_X = \{(Ff(x), x') \in FY \times FX \mid x \sqsubseteq_X x'\}$.

De hecho, según [HJ04], la condición (2.2) es equivalente a que (a) F sea estable y (b) para toda relación $R \subseteq X \times Y$,

$$\mathrm{Rel}(F)(R) \circ \sqsubseteq_X \ \subseteq \ \sqsubseteq_Y \circ \mathrm{Rel}(F)(R). \tag{2.3}$$

Otra forma de modelizar las simulaciones desde el punto de vista coalgebraico es por medio de los *relatores* definidos en la tesis de Thijs [Thi96]. Un *relator* para F se define como un endofuntor $\Gamma : \mathbf{Rel} \to \mathbf{Rel}$, tal que para toda relación $R \subseteq X \times Y$, se tiene $\Gamma(R) \subseteq FX \times FY$. En realidad las simulaciones requieren del uso de una clase especial de relatores, los denominados *relatores monótonos*.

Definición 2.2.17. *Dado un endofuntor $T : \mathbf{Sets} \to \mathbf{Sets}$, un* T-relator monótono *es un endofuntor $\Gamma : \mathbf{Rel} \to \mathbf{Rel}$ tal que*

- $\mathsf{U} \circ \Gamma = (T \times T) \circ \mathsf{U}$*, donde $\mathsf{U} : \mathbf{Rel} \to \mathbf{Sets} \times \mathbf{Sets}$ es el funtor olvidadizo,*
- $=_{TX} \subseteq \Gamma(=_X)$*, y*

[4] En la siguiente igualdad, la inclusión $\subseteq$ es siempre cierta, por lo que podríamos habernos limitado simplemente a pedir la contraria.

[5] De nuevo, la inclusión contraria es trivial.

- $\Gamma(S \circ R) = \Gamma(S) \circ \Gamma(R)$.

Entonces una Γ-*simulación* entre las coálgebras c y d es una Γ-coálgebra de la forma $(R, (c, d))$, es decir, una relación R tal que xRy implica $c(x)\Gamma(R)d(y)$.

De nuevo, como sucedía con las distintas nociones de bisimulaciones coalgebraicas, resulta que las nociones de simulación de Hughes y Jacobs y la de Thijs son, esencialmente, equivalentes. Decimos esencialmente, pues no es cierto que toda simulación definida por el formalismo de Hughes y Jacobs tenga su equivalente definida por un relator monótono; pero si nos limitamos a trabajar con los *órdenes estables*, sí que ambas nociones resultan ser equivalentes [Cîr06].

Esta relación entre los dos conceptos citados es muy estrecha, como se demuestra en [Thi96], ya que el *alzamiento de relaciones* se corresponde con el denominado *relator mínimo* inducido por un endofuntor T definido como $\Gamma_T = \langle T\pi_1, T\pi_2\rangle(TR) \subseteq TX \times TY$. Además, la simulación definida por el relator Γ_T coincide con la bisimulación [Cîr06].

2.2.4. Lógica temporal asociada a una coálgebra

Vamos a terminar este resumen de las definiciones coalgebraicas que hemos ido utilizando a lo largo de nuestra investigación, explicando cómo se pueden definir lógicas temporales asociadas a las coálgebras.

Si bien los programas secuenciales siguen un único camino desde su estado inicial hasta llegar a uno final, en el caso de los programas concurrentes ni tan siquiera podemos hablar en general de que tengan un estado final. Por ello, mientras que la lógica de Hoare (con su esquema de precondición/postcondición) es la herramienta de verificación por antonomasia de los programas secuenciales, a la hora de tratar con programas concurrentes hace falta otra herramienta.

Las *lógicas modales* son una rama de la lógica en la que se distinguen las nociones de *posibilidad* y *necesidad*, para así poder expresar aserciones del tipo "es posible que" o "necesariamente debe ocurrir". Las *lógicas temporales* son un caso particular de estas lógicas modales y fueron usadas en 1977 por Pnueli [Pnu77, MP92] para razonar sobre sistemas de transiciones reactivos. Por ello es natural que ya que las coálgebras generalizan a los sistemas de transiciones, se haya trabajado en lógicas temporales para ellas.

El primero en definir lógicas temporales para coálgebras fue Lawrence Moss, que en su trabajo de 1997, pero publicado en 1999 [Mos99], deducía a partir de cada funtor un lenguaje modal generalizado. Como comenta Martin Rößiger en [Röß00], el lenguaje de Moss era lo suficientemente expresivo como para distinguir fórmulas salvo bisimulación. Además, construía fórmulas características para los elementos de las coálgebras que expresaban su comportamiento futuro. Baltag [Bal00] siguió este camino para definir lógicas modales infinitarias que permitiesen capturar la bisimulación y la simulación.

El acercamiento que Bart Jacobs propone en el tutorial [Jac] es un modo relativamente sencillo de definir una lógica LTL asociada a las coálgebras. La peculiaridad de este enfoque es que Jacobs no usa proposiciones atómicas, sino que la lógica se sustenta en los denominados *predicados*. Para Jacobs un predicado sobre un conjunto, X, es sencillamente un subconjunto $P \subseteq X$. Para trabajar con estos se vuelve a echar mano del método del *alzamiento* en este caso aplicado a los predicados. Sin embargo, el mismo se define recurriendo al alzamiento de relaciones:

$$\mathrm{Pred}(F)(P) = \coprod_{\pi_1}(\mathrm{Rel}(F)(\coprod_{\delta}(P))) = \coprod_{\pi_2}(\mathrm{Rel}(F)(\coprod_{\delta}(P))),$$

donde $\delta = \langle id, id \rangle$, de manera que en concreto se tiene:

$\coprod_{\delta_x}(P) = \{(x, x) \mid x \in P\}$,

$\coprod_{\pi_1}(R) = \{x_1 \mid \exists x_2 . x_1 R x_2\}$ es el dominio de la relación R, y

$\coprod_{\pi_2}(R) = \{x_2 \mid \exists x_1 . x_1 R x_2\}$ es el codominio de R.

El concepto análogo a la bisimulación en términos de predicados es lo que se conoce como *predicado invariante*. Un predicado P es invariante para una coálgebra c, si cuando un elemento x pertenece a él, su sucesor $c(x)$ pertenece, en este caso, al alzamiento del predicado.

La sintaxis de la lógica LTL de Jacobs es la siguiente:

$$\varphi = P \subseteq X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi \,\mathcal{U}\, \varphi$$

donde, como es usual, a los operadores $\bigcirc$, $\Diamond$, $\Box$ y $\mathcal{U}$ se les llama *siguiente, alguna vez, siempre* y *hasta que*. La semántica de los operadores está definida de la siguiente manera:

- Para un predicado $P \subseteq X$, su semántica es $[\![P]\!] = P$.
- $[\![\bigcirc P]\!] = c^{-1}(\mathrm{Pred}(F)(P)) = \{x \in X \mid c(x) \in \mathrm{Pred}(F)(P)\}$.
- $[\![\Box P]\!]$ se define como el mayor invariante contenido en P, es decir, usando el operador $\nu : \mathcal{P}X \to \mathcal{P}X$ de *mayor punto fijo* se tiene $[\![\Box P]\!] = \nu S.(P \wedge \bigcirc S)$.
- $[\![\Diamond P]\!] = \neg\Box\neg P$.
- $[\![P \,\mathcal{U}\, Q]\!] = \mu S.(Q \vee (P \wedge \neg \bigcirc \neg S))$, donde $\mu : \mathcal{P}X \to \mathcal{P}X$ es el operador de *menor punto fijo*.

Diremos que un elemento x satisface una fórmula φ, denotado por $x \models \varphi$, cuando $x \in [\![\varphi]\!]$.

Pese a la sencillez de esta lógica su mayor pega es que no dispone de proposiciones atómicas, y que la base de las fórmulas son los predicados, es decir, subconjuntos de un X concreto. Esta carencia de proposiciones atómicas implica que no existe un asidero común a la hora de comparar fórmulas entre coálgebras sobre distintos espacios de estados, pues no existe una relación clara entre los predicados sobre X e Y, mientras que, por ejemplo la proposición atómica `true` tiene un significado común a cualquier espacio de estados.

Como se verá en la publicación `C1` que abre el capítulo 5 (página 87), se pueden incluir proposiciones atómicas en la lógica de Jacobs utilizando las ideas que Alexander Kurz empleó a lo largo de sus trabajos [Kur98, Kur00]. En concreto, las proposiciones atómicas se definen por medio de una transformación natural.

Otra aportación interesante es la de Corina Cîrstea [Cîr06], que presenta un método general para definir lógicas modales que caractericen las nociones de simulación definidas por relatores. El primer paso de esta construcción es definir la sintaxis del lenguaje

mediante el denominado *constructor del lenguaje*, definido como un endofuntor accesible[6] $\mathsf{S} : Alg(\Sigma_B) \to Alg(\Sigma_B)$. A Σ_B se le llama signatura y el *lenguaje* $\mathcal{L}(\mathsf{S})$ inducido por S es el álgebra inicial de S. En los casos más interesantes se tiene $\mathcal{L}(\mathsf{S}) = \bigcup_n L_n(\mathsf{S})$, con $L_0(\mathsf{S})$ el Σ_B-álgebra inicial y $L_{n+1}(\mathsf{S}) = \mathsf{S}(L_n(\mathsf{S}))$.

La semántica de la lógica se define recurriendo a límites de coálgebras, por lo que su construcción se realiza partiendo de una *semántica de un paso*. Para ello, en primer lugar, en [Cîr06] se define la *interpretación* de una Σ_B-álgebra L *sobre un conjunto* X como un morfismo de Σ_B-álgebras $d : L \to \mathcal{P}X$. Una interpretación indica para cada operador en la sintaxis los elementos de un conjunto X que lo satisfacen. En concreto, $x \in d(\varphi)$ significa que la fórmula φ se cumple en x. Las interpretaciones definen una categoría denotada por $\mathbf{Int}_B$, con $\mathsf{L} : \mathbf{Int}_B \longrightarrow Alg(\Sigma_B)$ el funtor que lleva d a L y $\mathsf{E} : \mathbf{Int}_B \longrightarrow \mathbf{Sets}^{op}$ el funtor que lleva d a X.

La *semántica de un paso* se define del siguiente modo:

Definición 2.2.18 ([Cîr06])**.** *La T-semántica de un constructor de lenguaje S es un funtor $\mathbb{S} : \mathbf{Int}_B \to \mathbf{Int}_B$ tal que $\mathsf{L} \circ \mathbb{S} = \mathsf{S} \circ \mathsf{L}$ y $\mathsf{E} \circ \mathbb{S} = T^{op} \circ \mathsf{E}$. Por consiguiente, una T-semántica para S lleva una interpretación $d : L \to \mathcal{P}X$ a otra interpretación $d' : \mathsf{S}L \to \mathcal{P}TX$.*

Finalmente, como decíamos antes, la lógica "final" se construye como límite de la lógica de un paso de acuerdo con la siguiente definición:

Definición 2.2.19 ([Cîr06])**.** *Para cualquier ordinal α, dada $(Z_\alpha), (\rho^\beta_\alpha : Z_\alpha \to Z_\beta)_{\beta \leq \alpha}$, la secuencia final del funtor T, la interpretación $d : \mathcal{L} \to \mathcal{P}Z_\alpha$ induce la lógica $(\mathcal{L}, \models)$ para T-coálgebras*

$$c \models_\gamma \varphi \text{ if and only if } \gamma_\alpha(c) \in d(\varphi),$$

donde $(\gamma_\alpha : C \to Z_\alpha)$ denota el cono sobre la secuencia final de T defina como:

- $\gamma_0 : C \to 1$ *es el único morfismo posible.*
- $\gamma_\alpha = T\gamma_\beta \circ \gamma$.
- γ_ω *es el único morfismo que satisface $\rho^\omega_\alpha \circ \gamma_\omega = \gamma_\alpha$ para cada $\alpha < \omega$.*

En particular, si la secuencia final de $\Gamma : \mathbf{Rel} \to \mathbf{Rel}$ se estabiliza en α[7], entonces la lógica inducida por $\mathbb{S}$ y Γ es la lógica inducida por la interpretación $d_\alpha : L_\alpha \to \mathcal{P}Z_\alpha$.

Para que la T-semántica $\mathbb{S}$ induzca la Γ-similitud es necesario que se cumpla la condición técnica de que S *preserva la expresividad* (véase [Cîr06] para los detalles concretos). En este caso, si además la secuencia final T se estabiliza en α y la secuencia inicial de S se estabiliza en α entonces, la lógica inducida por $\mathbb{S}$ y Γ caracterizan la relación de similitud [Cîr06, corolario 60].

[6] Un funtor es *accesible* si preserva para cierto cardinal infinito κ colímites κ-filtrados (en inglés κ-*filtered colimits*). Un colímite κ-filtrado de un funtor $F : J \to C$ es un colímite donde J es una categoría κ-filtrada, es decir, una categoría donde para todo diagrama $d : D \to E$ de cardinal menor que κ existe un cono sobre d [Mac98, BW99].

[7] Una secuencia final $(Z_\alpha), (\rho^\beta_\alpha : Z_\alpha \to Z_\beta)_{\beta \leq \alpha}$ se estabiliza en κ si $\rho^{\kappa+1}_\kappa$ coincide con la identidad.

Capítulo 3

Resultados

El presente capítulo está dedicado a la discusión de las principales contribuciones de las publicaciones que han dado como resultado esta tesis. Como comentábamos en la página 7, esta investigación empezó en el año 2005 con mi trabajo de tercer ciclo, titulado "Un estudio coalgebraico de las simulaciones" [Fáb06].

A lo largo de este tiempo hemos hecho un estudio de las simulaciones coalgebraicas, profundizando en cómo estas podían servir como marco unificado para el estudio de las semánticas de procesos. Este estudio no ha sido fácil y, como suele ser habitual al realizar una tesis, nos hemos ido enfrentando a distintas dificultades no previstas de antemano. La primera de ellas fue la necesidad de profundizar por mi parte en el estudio de los conceptos centrales de la teoría de categorías. La segunda aparecía ya en la propia definición de simulación coalgebraica, que desde el mismo inicio de la investigación nos impidió extender de manera completa el resultado clásico que dice que las simulaciones preservan las propiedades lógicas. Nuestras disensiones con la definición coalgebraica de simulación no se centraron únicamente en la complejidad de los resultados de preservación de propiedades, sino que definitivamente deseábamos garantizar que por medio de la misma se definiera un preorden de similitud. Ello solo queda garantizado cuando se cumple la condición de *estabilidad*, pero esta no es habitualmente sencilla de probar, por lo que Hughes y Jacobs sugieren quedarse con los órdenes que cumplen una condición más fuerte que nosotros llamamos *estabilidad por la derecha*.

Sin embargo nosotros pronto descubrimos que la generalidad de la noción original de simulación de Hughes y Jacobs, que *a priori* complicaba su estudio, podía servirnos sin embargo como marco unificador de distintas semánticas de procesos y de los distintos tipos de sistemas de transiciones que se usan para sustentarlas, sumergiéndonos así, como veíamos en la sección 1.4.1, en la consecución de nuestro objetivo 2.1.

De este modo, unificamos los sistemas probabilísticos y los sistemas de transiciones en el modelo de los sistemas de multitransiciones. Mientras tanto, seguíamos enfrascados en comprender las dificultades de la definición general de simulación. Recuerdo varias conversaciones por aquel entonces con David, en las que discutíamos si realmente Hughes y Jacobs necesitaban usar "dos veces" el orden $\sqsubseteq$ en la definición de simulación, pues al menos en los casos sencillos a nosotros nos parecía claro que bastaba con utilizarlo en un

único lado.

En esta línea fuimos profundizando en el estudio del concepto de *estabilidad*, y llegamos a entender cómo, ciertamente, en la mayoría de los casos previamente estudiados podíamos en efecto desprendernos del uso de uno de dichos órdenes. Sin embargo, vimos también que esto no era cierto en todos los casos interesantes, pues pronto dimos con dos nuevas nociones de simulación que necesitaban del uso combinado de los dos órdenes. Este estudio tuvo como resultado el descubrimiento de generalizaciones de la condición de *estabilidad por la derecha*, que daban lugar a simulaciones con propiedades que seguían siendo totalmente satisfactorias. Aplicando las mismas pudimos introducir en el marco coalgebraico abstracto las citadas nuevas nociones de simulación. Con ello dimos por concluida nuestra investigación en la presente tesis en relación con el objetivo 1, y pasamos a centrarnos en el objetivo 2.2 estudiando en profundidad las nuevas semánticas que habíamos introducido.

En concreto, estas dos nuevas nociones de simulación eran la simulación *covariante-contravariante* (en inglés *covariant-contravariant*) y la simulación *conforme* (en inglés, *conformance*). ¿Podíamos trasladar el estudio clásico recogido, por ejemplo, van Glabbeek [vG01], a estas nuevas semánticas? En particular, ¿podíamos caracterizarlas lógica y axiomáticamente? Trabajando ya únicamente con procesos en BCCSP, abordamos estas dos caracterizaciones, aunque, como veremos en la página 69, la axiomatización de la simulación covariante-contravariante nos supuso más de un quebradero de cabeza.

Nuestro trabajo pasó a estar entonces positivamente influenciado por la colaboración con Luca Aceto y Anna Ingólfsdóttir, ambos de la Universidad de Reykjavik (Islandia). Ellos sugirieron que explorásemos los probables puntos en común entre nuestra noción de simulación covariante-contravariante y el refinamiento modal. Este estudio nos llevó a trasladar los resultados del trabajo de Boudol y Larsen [BL92] al marco de los sistemas covariantes-contravariantes.

Tras este breve repaso cronológico de la investigación que ha dado lugar a las siete publicaciones que aparecen en la sección 5, a continuación pasamos a explicar con mayor detalle los resultados más importantes que aparecen en los citados artículos. En aras de una mayor claridad, en lugar de seguir con el orden cronológico, vamos a clasificar los resultados de forma temática de acuerdo con los objetivos que nos proponíamos en la sección 1.4.1. En concreto, en "El estudio categórico de la simulación" 3.1 trataremos la búsqueda de la satisfacción del objetivo 1, mientras que en "La unificación dentro del mundo categórico" 3.2 hablaremos de cómo alcanzamos el objetivo 2.2. Finalmente, en "El estudio de las nuevas semánticas de simulación" 3.3, explicaremos todo nuestro trabajo relacionado con el objetivo 2.2.

Las definiciones, teoremas y figuras que aparecen a lo largo de las próximas tres secciones están extraídas de los artículos originales, manteniendo su numeración original. No obstante, para mejorar su integración con el texto las hemos traducido al castellano.

3.1. El estudio categórico de la simulación

Como dijimos antes, iniciamos esta investigación durante mi trabajo de tercer ciclo. Animados por el tutorial de Jacobs sobre coálgebras [Jac], nos propusimos estudiar las

lógicas adecuadas para describir propiedades de los sistemas coalgebraicos. Nuestra primera meta fue la generalización del resultado de la tesis de Benthem que vimos en la sección 2.1.2: las bisimulaciones reflejan y preservan las propiedades definidas por una lógica temporal, mientras que las simulaciones (clásicas) preservan tales propiedades. Nosotros nos propusimos estudiar las condiciones bajo las cuales dichas propiedades eran preservadas (o reflejadas) de la forma oportuna, tanto por las bisimulaciones como por las simulaciones categóricas o sus variantes.

Partiendo de la base que nos proporcionaron los resultados que obtuvimos en el trabajo de tercer ciclo [Fáb06], nos propusimos retocarlos y pulirlos para obtener un artículo que pudiéramos mandar a un congreso internacional. El resultado de tal proceso de mejora es la publicación C1, que abre la sección 5. En este artículo quedaron de manifiesto esa serie de dificultades que comentábamos antes y que estaban causadas tanto por la definición categórica de simulación como por la de las lógicas empleadas para expresar las propiedades de las coálgebras. Por un lado, como hemos comentado en la sección 2.2.3, la definición propuesta por Hughes y Jacobs es demasiado general, y esta generalidad permite dos comportamientos no deseados. El primero es que se permite establecer relaciones de simulación que, intuitivamente, parecen que no debieran serlo, como es el caso de la *simulación lexicográfica*. El segundo, si bien no es una pega tan grave como la no transitividad, sí que es un hecho desconcertante que complica el estudio de las simulaciones y que hace que perdamos la intuición del sentido (de izquierda a derecha) de la simulación.

Por otro lado, decíamos, otro problema que debimos afrontar fueron las definiciones de las lógicas temporales empleadas para expresar las propiedades de los sistemas coalgebraicos. En un principio, ya que estábamos usando los conceptos de bisimulación y simulación de Jacobs, afrontamos este problema con las definiciones de la lógica temporal dadas por el mismo autor. Esta tenía la gran ventaja de obedecer a una intuición sencilla y natural; además al estar, al igual que la bisimulación, definida utilizando el concepto de alzamiento (pero en este caso de predicados), parecía un contexto más que adecuado para llegar a nuestra meta.

Por el contrario, como comentábamos en la página 51, los predicados sobre distintos conjuntos no tienen una relación clara. Para subsanar tal defecto relacionamos los elementos de X e Y por medio de relaciones y de sus *imágenes directas* e *inversas*. Sus definiciones, tal y como aparecern en C1 son las siguientes:

> *Dado un predicado P sobre X y una relación binaria $R \subseteq X \times Y$, diremos que $y \in Y$ está en la* imagen directa *de P (denotado como $y \in RP$), si existe $x \in X$ con $x \in P$ y xRy. La* imagen inversa *de P es la imagen directa para la relación R^{-1}.*

A partir de las imágenes directas e inversas definimos cómo se convertía una fórmula de la lógica de Jacobs sobre un conjunto X en otra sobre un conjunto Y, y viceversa. En concreto, si φ es una fórmula sobre X (respectivamente, sobre Y) formada sobre predicados P_i, su imagen φ^* sobre Y (respectivamente, su inversa φ^{-1} sobre X) se construye substituyendo los predicados P_i por sus imágenes directas RP_i (respectivamente, por sus imágenes inversas $R^{-1}P_i$). La siguiente definición recoge qué entendemos por *reflexión* y *preservación* en este contexto.

Definición 2 (extraído de C1.) Sea $R \subseteq X \times Y$ una relación binaria y a y b elementos tales que aRb. Decimos que R *preserva* la propiedad φ sobre X si, siempre que $a \models \varphi$ entonces, $b \models \varphi^*$. Diremos que R *refleja* la propiedad φ sobre Y si $b \models \varphi$ implica $a \models \varphi^{-1}$.

Las propias definiciones de reflexión y preservación de las lógicas ya nos indicaba que no iba a ser posible generalizar por completo el resultado clásico ni tan siquiera para el caso de las bisimulaciones, pues no podíamos trabajar con la negación. La negación no es un operador lógico monótono y, como se observa en la definición 2 de C1, nuestros conceptos de reflexión y preservación necesitan de tal monotonía. Tras una serie de lemas técnicos, llegábamos a que las bisimulaciones reflejaban y preservaban las propiedades lógicas construidas sin negación.

Inspirándonos en la lógica modal propuesta por Alexander Kurz [Kur98, Kur00], definimos una lógica modal con proposiciones atómicas y, ahora sí, con negación.

Definición 9 (extraído de C1.) Sea F : **Sets** $\to$ **Sets** un funtor y AP un conjunto de proposiciones atómicas. Sean también $\nu : F \Rightarrow \mathcal{P}(AP)$ una transformación natural y $c : X \to FX$ una coálgebra. Diremos que x satisface la proposición atómica $p \in AP$, y lo denotaremos por $x \models p$, cuando se cumpla $p \in (\nu_X \circ c)(x)$. De este modo $[\![p]\!] = \{x \mid p \in (\nu_X \circ c)(x)\}$.

Como recoge la definición 9 de C1, la semántica de las proposiciones atómicas se obtiene recurriendo a una interpretación natural. Así, obteníamos que dada una bisimulación entre x e y entonces, $x \in [\![\varphi]\!]_X$ si, y solo si, $y \in [\![\varphi]\!]_Y$, generalizando el resultado clásico para bisimulaciones.

Para el caso de las simulaciones de Hughes y Jacobs obtuvimos resultados parciales, puesto que para poder transferir las propiedades lógicas de una coálgebra a otra fue necesario imponer una serie de restricciones adicionales sobre los funtores y los órdenes participantes en el concepto de simulación, ya que, en general, un orden categórico cualquiera no permite relacionar las fórmulas lógicas entre coálgebras.

En primer lugar, como en el caso de las bisimulaciones, trabajamos con la lógica de Jacobs sobre predicados. Como primer intento usamos el concepto de orden *down-closed*, que ya se definía en [HJ04], y que dice que si se tienen dos elementos relacionados $a \sqsubseteq b$ entonces $b \in \mathrm{Pred}(F)(P)$ implica $a \in \mathrm{Pred}(F)(P)$. Restringiéndonos a estos órdenes demostramos que las simulaciones reflejan propiedades lógicas construidas únicamente con los operadores $\vee$, $\wedge$, $\bigcirc$ y $\Box$. Del mismo modo que nos habíamos centrado en los órdenes *down-closed* podíamos considerar la condición opuesta y definir los órdenes *up-closed* tales que si $a \sqsubseteq b$ entonces $a \in \mathrm{Pred}(F)(P)$ implica $b \in \mathrm{Pred}(F)(P)$. Para estos órdenes demostramos la preservación de propiedades pero, de nuevo, no para todos los operadores modales de la lógica.

Para demostrar un resultado que fuese válido para todos los operadores lógicos, vimos que necesitábamos además restringir la clase de los funtores. En nuestro caso, definimos la subclase **Order**, inspirándonos en la clase **Poly** definida en [HJ04]. La idea es trabajar únicamente con funtores polinomiales y con órdenes que se descompongan adecuadamente según la estructura de los funtores polinomiales. Es decir, es indispensable trabajar con órdenes cuya definición particular tenga en cuenta la estructura del funtor sobre el que

están definidos. Por ejemplo, sobre el funtor producto considerábamos únicamente los órdenes que fueran a su vez producto de dos subórdenes. Para estos funtores y órdenes de **Order** demostramos que las simulaciones reflejan, y también preservan, las propiedades lógicas.

Finalmente, si incluimos las proposiciones atómicas en la lógica, necesitamos no solo restringir los resultados a la clase **Order**, sino imponer una condición adicional al orden para que este permita trasladar adecuadamente el conjunto de proposiciones atómicas definida mediante la transformación natural que veíamos en la definición 9 de C1. De este modo definimos la clase **Down-Natural ν-Order** como la subclase de **Order** cuyos órdenes son *down-natural*:

> **Definición 10** (extraído de C1.) Sea $F : \mathbf{Sets} \to \mathbf{Sets}$ un funtor, AP un conjunto de proposiciones atómicas y $\nu : F \Rightarrow \mathcal{P}(AP)$ una transformación natural. Decimos que $\sqsubseteq$ es un ν-orden *down-natural* si siempre que tengamos $u \sqsubseteq u'$ también se tiene $\nu(u') \subseteq \nu(u)$.

A partir de la definición 10 de C1 obtuvimos la generalización del resultado clásico, es decir, si una simulación coalgebraica relaciona x e y entonces

$$y \in [\![\varphi]\!]_Y \quad \Longrightarrow \quad x \in [\![\varphi]\!]_X \,.$$

Análogamente, considerando el concepto opuesto de órdenes *up-natural*, para los que basta considerar $\nu(u) \subseteq \nu(u')$ en la definición 10 de C1, obtuvimos el resultado de preservación de propiedades.

Estos resultados parciales fueron los que nos hicieron darnos cuenta de la necesidad de profundizar en el estudio del concepto coalgebraico de simulación, tratando de obtener ideas que pudieran valer para justificar, entre otras cosas, cuál era el sentido canónico en la simulación coalgebraica. Aquella búsqueda continuó donde empezó, es decir, en el artículo de Hughes y Jacobs; y todos los resultados que a continuación comentamos aparecen en la publicación C2.

Comentábamos en la sección 2.2.3 que para garantizar la transitividad de la similitud los autores proponían quedarse únicamente con aquellos órdenes que eran *estables por la derecha*. Los autores de [HJ04] argüían que la necesidad de introducir la *estabilidad por la derecha* respondía a que era sensiblemente más sencilla de comprobar que la *estabilidad*. Además, la inmensa mayoría de los órdenes concretos que manejaban eran en efecto *estables por la derecha*. Pero, lo que no se explicaba ni motivaba por ninguna parte era el porqué de tal sesgo "hacia la derecha", ni se llegaba al fondo en la relación con el hecho de que una propiedad más restrictiva fuera la llave razonable para alcanzar la propiedad (¡simétrica!) de estabilidad.

Nosotros llegamos a una reveladora conclusión: de la condición (2.3) de la página 49, se puede obtener también que

$$\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X \;=\; \sqsubseteq_Y \circ \mathrm{Rel}(F)(R). \tag{3.1}$$

De esta manera, si una simulación es *estable por la derecha* para un orden $\sqsubseteq$, en vez de usar la definición general, se puede usar de manera equivalente la igualdad 3.1 definida arriba. Así, el orden $\sqsubseteq$ puede usarse únicamente en uno de los lados de la definición, por lo

que la verificación de propiedades se vuelve más sencilla que cuando usamos directamente la definición original.

Como comentábamos en la página 47 de la sección 2.2.3, en general, la potencia de los dos órdenes a manejar no es equiparable pues de hecho hay situaciones en las que uno de ellos no puede usarse como se deseara. Nos servirá como ejemplo el caso de la simulación estándar entre LTS no etiquetados que comentábamos en la citada sección, formalizado en la forma:

$$c(x) \subseteq_X u \operatorname{Rel}(\mathcal{P}id)(R)\ v \subseteq_Y d(y)\,. \tag{3.2}$$

En la ecuación de arriba el orden $\subseteq_Y$ indica que el proceso definido por la coálgebra d puede "deshacerse" de algunas transiciones para bisimular al proceso definido por la coálgebra c. Esta interpretación está en correspondencia directa con la idea clásica de que un proceso q simula a otro p siempre que q sea capaz de hacer todo lo que p puede hacer y posiblemente algo más. Del mismo modo, se podría considerar en principio que al igual que q puede deshacerse de las transiciones "extras" que le sobran, p podría añadirse aquellas transiciones que le faltan para hacer lo que hace q. Esto es en efecto lo que podría eventualmente conseguirse utilizando a la izquierda el orden $\subseteq_X$. Pero para ello sería necesario que el espacio X contuviera los suficientes elementos, lo que no es siempre cierto.

La simulación estándar es *estable por la derecha* y por tanto, también *estable*. Esto se traduce en que podemos prescindir del orden $\subseteq_X$, y usar únicamente la potencia del orden $\subseteq_Y$. ¿Pero, qué ocurriría si consideramos la simulación "contraria", es decir, la relación de anti-simulación o de "ser simulado por"? En ese caso tenemos la siguiente ecuación:

$$c(x) \supseteq_X u \operatorname{Rel}(\mathcal{P}^A id)(R)\ v \supseteq_Y d(y)\,. \tag{3.3}$$

Ahora, el orden que nos da toda la potencia necesaria es $\supseteq_X$ que, de nuevo, nos permite desprendernos de transiciones. Se ve de este modo que la anti-simulación difícilmente será *estable por la derecha*, pues el papel que juega aquí el orden $\supseteq_Y$ a la derecha no nos permite demostrar, en general, si dos procesos son anti-similares. Pero es evidente que estando definida la anti-simulación por el preorden opuesto al que caracteriza a la simulación, la misma hereda todas las buenas (y las malas, en su caso) propiedades de su opuesto. En concreto, la composición de anti-simulaciones es también una anti-simulación, y por tanto la anti-similitud es un preorden. De hecho, un resultado casi inmediato que probamos es que si un orden es estable, entonces su opuesto también lo es.

Así llegamos a una primera generalización que dedujimos a partir de la definición de *estabilidad por la derecha* y el ejemplo de la anti-simulación: del mismo modo que Hughes y Jacobs se habían centrado en la *estabilidad por la derecha*, podrían haberlo hecho sobre la condición simétrica, es decir, sobre la *estabilidad por la izquierda* que se recoge en la siguiente definición.

Definición 2 (extraído de C2.) Decimos que un funtor F con orden $\sqsubseteq$ es *estable por la izquierda* si, para toda función $f : X \to Y$, se tiene

$$(Ff \times id)^{-1} \sqsubseteq_Y \ \subseteq \coprod_{id \times Ff} \sqsubseteq_X .$$

Una vez más rige el principio, lo bello y elegante suele ser simétrico. La anti-simulación es *estable por la izquierda* y, es más, un orden $\sqsubseteq$ es *estable por la derecha* si, y solo si, el orden opuesto $\sqsubseteq^{op}$ es *estable por la izquierda*. De este modo obtuvimos una respuesta a nuestra pregunta de cuál debía ser la dirección canónica para la simulación.

- Si queremos considerar únicamente simulaciones que tengan el mismo sentido de la simulación estándar, es decir, relaciones tales que cuando se tenga $p \lesssim q$ ello indique que q tiene "más comportamientos posibles" que p, entonces el sentido canónico viene dado por la propiedad de *estabilidad por la derecha*.

- Si, en cambio, estamos interesados en aquellas simulaciones cuyo sentido es el de la anti-simulación, en este caso, relaciones tales que $p \lesssim q$ indique que q tiene "menos comportamientos posibles" que p, entonces el sentido canónico viene dado por la propiedad de *estabilidad por la izquierda*.

La obtención del criterio anterior no supuso la finalización de nuestro estudio de la noción canónica de simulación. Apoyándonos en los conceptos de estabilidad por la derecha y por la izquierda, pudimos definir otras condiciones más generales que siguen siendo más fuertes que la *estabilidad* y que, como dice el título del artículo C2 en el que recogemos estos resultados, nos indica que hay nociones de simulación interesantes que no cumplen la condición de *estabilidad por la derecha*, pero tampoco la *estabilidad por la izquierda*.

La primera de estas generalizaciones corresponde a los órdenes *side estables*, que son los órdenes que se distribuyen con respecto a las acciones dando lugar a componentes que son bien *estables por la derecha* o *izquierda*, como se define a continuación.

> **Definición 5** (extraído de C2.) Decimos que un orden $\sqsubseteq$ sobre un funtor F^A es *distributivo sobre acciones* si existe una familia de órdenes $\sqsubseteq^a$ sobre F tales que
>
> $$f \sqsubseteq g \iff f(a) \sqsubseteq^a g(a) \text{ para todo } a \in A.$$
>
> Siempre que $\sqsubseteq$ se pueda distribuir de esta manera escribiremos $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$.
>
> **Definición 6** (extraído de C2.) Decimos que un orden distributivo sobre acciones $\sqsubseteq$ sobre F^A es *side stable* si en la descomposición $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$ se tiene que cada orden es estable por la derecha, o bien, estable por la izquierda.

El segundo resultado es, en cierto sentido, parecido al anterior, pero no necesita involucrar acciones, sino que se basa en la composición adecuada de órdenes estables:

> *Si un orden se puede obtener como composición de dos subórdenes que conmuten entre sí, siendo uno* estable por la derecha *y el otro* estable por la izquierda*, entonces el orden en cuestión es estable.*

La ventaja de los dos conceptos anteriores es que ambos permiten combinar las partes *estables por la derecha* y *estables por la izquierda* de los órdenes para obtener una condición análoga a la condición (3.1), en la que la parte $\sqsubseteq^r$ del orden correspondiente

que es *estable por la derecha* se usa en el lado derecho de la ecuación y la parte $\sqsubseteq^l$ que es *estable por la izquierda* en el izquierdo. Es decir, obtenemos la igualdad:

$$\sqsubseteq \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq \;=\; \sqsubseteq^l \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^r. \tag{3.4}$$

En el artículo C2, aparte de discutir todos estos detalles sobre la estabilidad que nos ayudan a comprender mejor el concepto de simulación y arrojar luz sobre cuál debe ser el *sentido canónico* del orden, definimos dos nuevos conceptos de simulación. Estos conceptos, que están estrechamente relacionados con los sistemas modales y el *testing*, fueros los que motivaron las definiciones de las generalizaciones de los *órdenes side-stable* y la composición de órdenes. Estas dos generalizaciones forman el primer pilar del puente que hay entre los objetivos 1 y 2.1, pues no dejan de ser un marco unificado de tratar órdenes más complejos, que como muestran las nuevas semánticas de simulación (páginas 63 y 69), nos permiten trabajar con simulaciones más complejas. Así, estas definiciones también relacionan el objetivo 1 con el 2.2 y, además, ilustran perfectamente este nexo de unión que existe entre las dos partes de nuestra investigación: el estudio desde el punto de vista más teórico y abstracto de las simulaciones categóricas, y la unificación de las semánticas de simulación.

En la publicación C3 hicimos un primer estudio con la doble motivación de comprender mejor la noción de simulación de Hughes y Jacobs, y entender cómo la maquinaria coalgebraica podía servir para unificar ciertas definiciones clásicas de simulación en distintos ámbitos (el objetivo 2.1). Este trabajo, que cierra el bloque de las publicaciones de corte categórico del capítulo 5, es además el otro pilar del puente entre los objetivos 1 y 2.1, pues en él no dejábamos de lado el estudio de las propiedades generales de la simulación. En concreto, el eje de esta publicación fue la manera en que las transformaciones naturales permitían trasladar (bi)simulaciones coalgebraicas entre diferentes estructuras.

Antes de enunciar el teorema más general del artículo necesitamos presentar dos conceptos que hacen uso de transformaciones naturales. El primero de ellos es la transformación de un orden funtorial en G en otro en F: dado un orden $\sqsubseteq$ en G, definimos el *orden inducido* por una transformación natural $\alpha : F \Rightarrow G$ como

$$x \sqsubseteq_G^{\alpha-} x' \iff \alpha_X(x) \sqsubseteq_G \alpha_X(x')$$

Podemos decir que el orden $\sqsubseteq_G^{\alpha-}$ en F es la representación canónica en F del orden $\sqsubseteq_G$ en G.

El segundo concepto es la representación de una coálgebra en G por otra en F: si tenemos una G-coálgebra $b : X \to GX$, decimos que $a : X \to FX$ es un *representante concreto* o un *F-representante* de b si, y solo si,

$$b = \alpha_X \circ a\,.$$

Igual que antes, b puede ser vista como la imagen canónica en F de una coálgebra en G.

El teorema en cuestión es general al tratar con simulaciones categóricas lo que, en particular, permite que sea válido también para la bisimulación. Su enunciado es el que se incluye en el siguiente teorema.

> **Teorema 4** (extraído de C3.) Sea $\alpha : F \Rightarrow G$ una transformación natural que sea un epimorfismo, $\sqsubseteq_G$ un orden en G y $b_1 : X_1 \to GX_1$, $b_2 : X_2 \to GX_2$

dos coálgebras, con $a_1 : X \to FX$, $a_2 : Y \to FY$ dos F-representantes arbitrarios. Entonces, las $\sqsubseteq_G$-simulaciones que relacionan b_1 y b_2 coinciden con las $\sqsubseteq_G^{\alpha-}$-simulaciones entre a_1 y a_2.

De nuevo, podemos interpretar el teorema 4 de C3 como un teorema de representación que permite relacionar de un modo canónico las F-simulaciones con las G-simulaciones. Además, en la página 62 presentaremos la aplicación que nosotros le hemos dado al resultado. En concreto, veremos cómo el teorema es nuestra principal herramienta para unificar diversas nociones (clásicas) de (bi)simulación, dentro de una misma F-coálgebra.

3.2. La unificación dentro del mundo categórico

Para abordar el estudio de la unificación de conceptos de simulación clásicos nos inspiramos en los ejemplos concretos de las bisimulaciones ordinarias y las bisimulaciones probabilísticas, y estudiamos la manera en la que ambas se podían definir dentro del marco común de bisimulaciones categóricas sobre multiconjuntos. Pero, como acabamos de comentar, al margen de esa aplicación concreta al mundo de los sistemas de transiciones (probabilísticos y estándar) también exploramos las ideas desde un punto de vista puramente categórico y obtuvimos resultados generales válidos para todas las simulaciones de Hughes y Jacobs.

Usamos como punto de partida la definición categórica formulada por Vink y Rutten (como vimos en la sección 2.2.2), en donde consideraban el funtor $\mathcal{D}$ de distribuciones probabilísticas. Pese a la elegancia que otorga esta caracterización, la principal pega que le achacábamos era que nos obligaba a abandonar el marco de los *sistemas de transiciones probabilísticos* (clásicos) para movernos en el marco abstracto de las *distribuciones de probabilidad*, por lo que la definición de Vink y Rutten no es la mejor a la hora de comparar resultados clásicos.

Es entonces cuando, para tratar de recuperar los *sistemas de transiciones probabilísticos*, y así poder comparar adecuadamente los resultados clásicos con los categóricos, decidimos modelizarlos con *sistemas de multitransiciones*, es decir, sistemas de transiciones en los que pueden aparecer varias transiciones idénticas o, en otras palabras, consideramos un multiconjunto de sucesores para cada estado, en lugar de un conjunto. Como veíamos en la sección 2.1.2, la definición clásica de los sistemas de transiciones probabilísticos no permitía distinguir entre varias formas distintas de llegar a un proceso después de ejecutar una acción. Nosotros considerábamos que esa limitación no era deseable.

Por ejemplo, si queremos definir la semántica operacional del proceso $p = \frac{1}{2}a + \frac{1}{2}a$, intuitivamente tendremos dos transiciones distintas que llegan al mismo estado final *stop*. Pero, si usamos la definición original de Larsen y Skou, estamos juntando ambas transiciones en una única, resultando $p \xrightarrow{a}_1 stop$. Si bien es cierto que siempre se pueden mantener ambas transiciones separadas recurriendo a dos estados finales $stop_1$ y $stop_2$ distintos, esto depende de la manera en que definamos el conjunto Pr y no parece una solución elegante, sino un mero "parche". En cambio, si trabajamos con multiconjuntos, esa distinción viene dada por defecto.

Modelizamos los sistemas de multitransiciones como coálgebras del funtor $\mathcal{M}(A \times X)$, y los sistemas de transiciones probabilísticos como coálgebras del funtor $\mathcal{M}_1([0,1] \times A \times X)$, donde únicamente consideramos multiconjuntos en los que la suma de las probabilidades (dadas por la primera componente) de todos sus elementos es 1. Además, las definiciones de los alzamientos para el funtor $\mathcal{M}$ y sus particularizaciones no son complicadas, lo que nos permitió mantener la simplicidad de la teoría.

Aunque la definición de los sistemas de transiciones probabilísticos como coálgebras del functor $\mathcal{M}_1([0,1] \times A \times X)$ es natural, no es perfecta. El mayor problema es que de este modo estamos considerando transiciones etiquetadas por pares del conjunto $[0,1] \times A$, en donde la primera componente no tiene ningún significado especial. Este problema se manifiesta en primer lugar al comparar la definición de la bisimulación probabilística con la bisimulación coalgebraica para el funtor $\mathcal{M}_1$ que vemos a continuación.

> **Definición 1** (extraído de C3.) Una bisimulación probabilística en una coálgebra $p : X \to \mathcal{M}_1([0,1] \times A \times X)$ es una relación de equivalencia $\equiv_p$ en X tal que, siempre que $x_1 \equiv_p x_2$, con $p(x_i) = \sum t_j^j \cdot (p_j^i, a_j^i, x_j^i)$, también se tiene $\sum\{t_j^1 \cdot p_j^1 \mid a_j^1 = a,\, x_j^1 \in E\} = \sum\{t_j^2 \cdot p_j^2 \mid a_j^2 = a,\, x_j^2 \in E\}$, $a \in A$ y toda clase de equivalencia E en $X/\equiv_p$.

Si consideramos la definición de bisimulación categórica entre coálgebras del funtor $\mathcal{M}_1([0,1] \times A \times \cdot)$, encontramos que dados $X = \{x\}$, $Y = \{y\}$, $p_a : X \to \mathcal{M}_1([0,1] \times A \times X)$ con $p_a(x) = 1 \cdot (1, a, x)$ y $p_b : Y \to \mathcal{M}_1([0,1] \times A \times Y)$ con $p_b(y) = 2 \cdot (\frac{1}{2}, a, y)$, entonces no hay bisimulación posible entre x e y. Esto quiere decir que las $\mathcal{M}_1([0,1] \times A \times \cdot)$-bisimulaciones no son equivalentes al concepto clásico de bisimulación probabilística, al contrario de lo que ocurre con las $\mathcal{D}$-bisimulaciones, como veíamos en la página 45 en la sección 2.2.2.

Para conseguir la deseada equivalencia usamos una transformación natural que nos permitiese relacionar estos dos tipos distintos de bisimulación. En concreto, consideramos la transformación natural $\mathcal{D}_{M_X} : \mathcal{M}_1([0,1] \times X) \to \mathcal{D}(X)$ entre nuestra clase especial de sistemas de multitransiciones y los sistemas de transiciones probabilísticos de Vink y Rutten.

Del mismo modo, podíamos usar el marco de los multisistemas de transiciones para trabajar con LTS estándar. Igual que antes, las bisimulaciones entre las coálgebras del funtor $\mathcal{P}(\cdot)$ y las coálgebras del funtor $\mathcal{M}(\cdot)$ no coinciden directamente, por lo que para relacionarlas adecuadamente usamos otra transformación natural, $\{\cdot\} : \mathcal{M} \Rightarrow \mathcal{P}$. Además, aprovechando que ya teníamos el marco común de los sistemas de multitransiciones, discutimos un modo de mezclar no determinismo y elecciones probabilísticas, en el que de un modo sencillo podíamos combinar ambos formalismos.

Como decíamos en la página 61 , estos resultados son una aplicación del teorema 4 de C3 en el que aprovechamos que la bisimulación es un caso particular de la simulación. En concreto, para los LTS, estamos considerando el funtor $F = \mathcal{M}(\cdot)$, $G = \mathcal{P}(\cdot)$ y la transformación $\alpha = \{\cdot\}$; mientras que para los sistemas probabilísticos consideramos $F = \mathcal{M}_1([0,1] \times \cdot)$, $G = \mathcal{D}(\cdot)$ con la transformación $\alpha = \mathcal{D}_{M_\cdot}$.

Por tanto, el teorema 4 de C3 puede usarse para unificar las G_1-coálgebras y las G_2-coálgebras como F-coálgebras, bajo el supuesto de que existan transformaciones naturales $\alpha_1 : F \Rightarrow G_1$ y $\alpha_2 : F \Rightarrow G_2$ que permitan relacionar sus funtores.

En consecuencia, en la publicación C3 hacíamos un estudio de las (bi)simulaciones y sus propiedades con un doble carácter: nos acercábamos un poco más a nuestro objetivo 1, e iniciábamos la unificación de las semánticas clásicas dentro del marco categórico y coalgebraico. Este segundo objetivo se complementaba con el estudio de nuevas semánticas de simulación que, en concreto, son las nociones que obteníamos en el artículo C2 y que, hasta ahora, únicamente hemos insinuado.

3.3. El estudio de nuevas semánticas de simulación

En la publicación C2 vimos también cómo de un modo natural se podían unificar en una misma relación de simulación las definiciones de simulación clásica, la anti-simulación y la bisimulación. A esta nueva semántica de simulación la llamamos *simulación covariante-contravariante*, debido a que el orden coalgebraico que la definía distinguía el tipo de las acciones del LTS, y estaba definido de manera que a un tipo de acciones le aplicaba la condición clásica de simulación (de ahí las acciones *covariantes*), mientras que al otro le aplicaba la condición opuesta (las *contravariantes*). Además, la simulación covariante-contravariante era la motivación para la definición general de aquellos órdenes *side-stable*, y por tanto, era una simulación con buenas propiedades.

Por otro lado, al tratar de encontrar una noción canónica de simulación que respondiese del modo más coherente posible a nuestra intuición, definimos la noción de *simulación conforme*, llamada así por nuestra inspiración en las técnicas de *testing*. Esta simulación considera que un proceso p es mejor que otro q siempre que p sea más determinista que q. De esta manera se obtiene un concepto que, creemos, está mucho más cerca de la intuición que el concepto clásico de simulación, como veremos más adelante (página 71).

El objetivo 2.2 forma parte del segundo gran tema de estudio de la tesis, es decir, del estudio de las semánticas de procesos desde el punto de vista más clásico, y se basa en la profundización en estos dos conceptos nuevos de simulación: la *covariante-contravariante* y la *conforme*. A lo largo de dos artículos fuimos reproduciendo los conceptos básicos necesarios para entender una noción de simulación: sus caracterizaciones lógicas y sus axiomatizaciones de acuerdo con el estándar de van Glabbeek. Además, y centrándonos ya únicamente en la noción de simulación covariante-contravariante (o abreviadamente cc-simulación), hemos estudiado más profundamente su relación con otros conceptos de simulación próximos a ella, así como sus fórmulas características y cómo estas representan procesos (lo que Boudol y Larsen denominaron en [BL92] *representación gráfica*).

La simulación covariante-contravariante

Empecemos viendo y discutiendo la primera de las dos nuevas nociones de simulación: las simulaciones covariantes-contravariantes.

Definición 3.3.1. *Dados* $c : X \to \mathcal{P}(X)^{Act}$ *y* $d : Y \to \mathcal{P}(Y)^{Act}$ *sistemas de transiciones con el alfabeto Act, y* $\{Act^r, Act^l, Act^{bi}\}$ *una partición de este, una* (Act^r, Act^l)-simulación *(o simplemente una* simulación covariante-contravariante*) entre c y d es una relación* $S \subseteq X \times Y$ *tal que para todo* $(x, y) \in S$ *se cumple:*

- *para todo* $a \in Act^r \cup Act^{bi}$ *con* $x \xrightarrow{a} x'$ *existe* $y \xrightarrow{a} y'$ *con* $(x', y') \in S$*, y*

- *para todo* $a \in Act^l \cup Act^{bi}$ *con* $y \xrightarrow{a} y'$ *existe* $x \xrightarrow{a} x'$ *con* $(x', y') \in S$.

Escribiremos $x \lesssim_{CC} y$ *si existe una simulación covariante-contravariante entre* x *e* y.

Como vimos en C2, las simulaciones covariantes-contravariantes pueden definirse como simulaciones categóricas.

> **Proposición 1** (extraído de C2.) Las (Act^r, Act^l)-simulaciones se pueden definir como las simulaciones coalgebraicas para el funtor $F = \mathcal{P}^{Act}$, con el orden funtorial ${}_A\sqsubseteq_{A'}$ tal que, para cada conjunto X y $\alpha, \alpha' : Act \to \mathcal{P}(X)$, se tiene $\alpha \; {}_A\sqsubseteq_{A'} \; \alpha'$ siempre que:
>
> - Para toda $a \in Act^r \cup Act^{bi}$, $\alpha(a) \subseteq \alpha'(a)$.
> - Para toda $a \in Act^l \cup Act^{bi}$, $\alpha(a) \supseteq \alpha'(a)$.

Como decíamos, la primera idea que nos llevó a definir las simulaciones covariantes-contravariantes fue la de unificar los conceptos de simulación estándar, anti-simulación y bisimulación. De hecho, obviamente, si $A = A^r$ entonces la cc-simulación coincide con la simulación estándar, mientras que si $A = A^l$ tenemos la anti-simulación y, finalmente, si $A = A^{bi}$, la cc-simulación coincide con la bisimulación. De ahí también vino la nomenclatura que usamos para los conjuntos de acciones: A^r son aquellas acciones que tienen una condición de *estable por la derecha*, A^l son las acciones de la condición de *estable por la izquierda* y A^{bi} son acciones bivariantes.

Tras la lectura de la definición, salta a la vista que nuestro concepto de simulación covariante-contravariante está ligado con las simulaciones entre sistemas con entrada y salida que veíamos en la sección 2.1.3: los autómatas con entrada/salida y los *interface automata*. En efecto, nosotros distinguimos entre tres tipos de acciones y por ello las simulaciones covariantes-contravariantes son también un medio adecuado para comparar sistemas que distingan entre acciones de entrada y de salida. En nuestro contexto, y de un modo parecido a la simulación alternante en el caso de los *interface automata*, las entradas son las acciones que deben comportarse según el sentido de la simulación estándar (las acciones en A^r) ya que, en principio, queremos que nuestro preorden permita el mayor número de comportamientos posibles; mientras que las salidas (las acciones en A^l), son aquellas acciones que deben comportarse en el sentido contrario al de la simulación estándar, pues como ya discutíamos en la sección 2.1.3, más salidas implica una mayor restricción en el entorno (en particular se debe disponer del canal adecuado para tratar cada una de las acciones de salida).

¿Pero, en qué lugar quedan nuestras acciones bivariantes? En primer lugar, estas acciones no se corresponden con las acciones internas de los *interface automata*. Nuestras acciones bivariantes no se ocultan al entorno, sino que, intuitivamente, deben verse como acciones con un doble carácter: pueden verse como acciones que son tanto botones de entrada como canales de salida; por ello, se comportan de acuerdo con la semántica de la bisimulación. Es cierto que la motivación aquí no resulta demasiado intuitiva y, como discutiremos más adelante (en la página 69), pese a que aparentemente las acciones bivariantes son aquellas que mejor deberían comportarse (pues están relacionadas con un concepto tan equilibrado como la bisimulación), aparecen diversos problemas con ellas cuando se usan en compañía con acciones covariantes o contravariantes.

Ya vimos (página 27) que Larsen junto con el resto de sus co-autores de [LNW07] encontró una relación entre los *interface automata* y los sistemas modales, por lo que era razonable plantearse si ocurriría lo mismo con la simulación covariante-contravariante. La respuesta es aún más satisfactoria que en el citado caso: en realidad existe una relación muy estrecha con los sistemas modales y sus refinamientos, como recientemente estudiamos en la publicación S1 de 2011.

En este artículo, como su propio título indica, estudiamos la relación entre la cc-simulación y la bisimulación parcial sobre LTS, y los refinamientos modales sobre MTS. En el primer caso llegamos a ver que la bisimulación parcial es sencillamente un caso particular de la cc-simulación. Mucho más interesante es la relación entre la cc-simulación y el refinamiento modal. El enfoque que utilizamos fue el mismo que el de Larsen para relacionar los *interface automata* con los sistemas modales. Así, construimos transformaciones para relacionar nuestros sistemas de transiciones módulo la cc-simulación con los sistemas modales módulo el refinamiento. En concreto, fuimos capaces de construir dos transformaciones: $\mathcal{M}$ que transformaba un LTS en un MTS y $\mathcal{C}$ que transformaba un MTS en un LTS. Estas transformaciones mantienen la cc-simulación y el refinamiento modal, así como la lógica covariante-contravariante (que veremos en la página 68) y modal.

Aunque la propia existencia de estas transformaciones conlleva ya más buenas propiedades, pone de manifiesto además que nuestra noción de cc-simulación es, en cierto sentido, equivalente a la noción de refinamiento modal sobre MTS, si bien el estudio categórico (en el marco de las instituciones) de ambas nociones reveló que esa relación no era realmente "uno a uno". Parafraseando el eslogan de Goguen y Burstall [GB92] podemos decir que la verdad que define la lógica modal de los MTS no es un mero renombramiento de la lógica covariante-contravariante.

En primer lugar, se puede decir que, de las dos transformaciones, $\mathcal{C}$ es la más intuitiva puesto que viene a decirnos que un MTS es un LTS módulo cc-simulación, considerando simplemente un cambio del nombre de las acciones: las acciones b que aparecen en las transiciones *may* pasan a ser $ct(b)$ (acciones contravariantes) y las acciones a que aparecen en una transición *must* pasan a ser $cv(a)$ (acciones covariantes). Como, además, por definición, las transiciones *must* son también en particular *may*, en realidad las acciones a que aparecen en una transición *must* se transforman en dos acciones $cv(a)$ y $ct(a)$, haciendo explícito de este modo el carácter bivariante que tienen las transiciones *must* en los MTS.

En cambio la transformación $\mathcal{M}$, es algo más farragosa y más cercana a la que se definía en [LNW07]. En concreto, necesita añadir al MTS transformado un estado extra u desde el que se pueden ejecutar transiciones *may* correspondientes a todas las acciones de A, que nos devuelven al propio estado u. La transformación incorpora por cada a-transición una a-transición *may*. Además, para cada a-transición con $a \in A^r \cup A^{bi}$ se añade una a-transición *must*. Finalmente, para cada transición $a \in A^r$ y estado p se añade una transición $p \xrightarrow{a}_\diamond u$. Gracias a la incorporación del estado adicional u junto con sus transiciones *may* queda neutralizado el carácter *may* que nos vemos obligados a incorporar al traducir las acciones de A^r.

Decíamos antes que explotamos las transformaciones entre LTS y MTS también dentro del marco categórico de las instituciones. Para ello, construimos instituciones $\mathcal{I}_{cc}$ para

caracterizar la lógica covariante-contravariante, e $\mathcal{I}_{mts}$ para la lógica modal. Hemos visto que podemos definir un *morfismo entre instituciones* de $\mathcal{I}_{mts}$ a $\mathcal{I}_{cc}$, es decir, que tenemos una manera de pasar la lógica modal a la lógica covariante-contravariante de un modo categórico que garantiza que esa traducción preserva la lógica. Una vez que se define dicho morfismo, era lógico que nos planteásemos si ello podía significar que, en realidad, $\mathcal{I}_{mts}$ era una subinstitución[1] de $\mathcal{I}_{cc}$. La respuesta a esta pregunta fue negativa ya que, de hecho, en el marco de los LTS sin acciones bivariantes existen tanto la implementación universal como la especificación universal; mientras que en el marco modal existe la especificación universal, pero no una implementación universal. Traducido al lenguaje categórico, esto significa que dentro de los modelos de los LTS módulo cc-simulación existe un objeto *débilmente final*[2], pero no así en los sistemas modales módulo refinamiento. Para el caso contrario conjeturamos que ni tan siquiera existe un *morfismo entre instituciones* de $\mathcal{I}_{cc}$ a $\mathcal{I}_{mts}$, lo que (entendemos) viene a justificar que la transformación $\mathcal{M}$ es menos natural que la transformación $\mathcal{C}$.

Una vez que tuvimos las transformaciones $\mathcal{C}$ y $\mathcal{M}$ decidimos estudiar con detalle las propiedades que se derivaban de ellas al componerlas. La más intuitiva de esas composiciones es $\mathcal{C} \circ \mathcal{M}$, que nos permitía pasar de un LTS T con acciones bivariantes a otro LTS $\mathcal{C} \circ \mathcal{M}(T)$ sin ellas. Este estudio se continuó y expandió en el artículo S2.

En esa publicación seguimos el modelo de [BL92] para, en primer lugar, construir la fórmula característica de un proceso, y después, estudiar el paso contrario: ver a partir de una fórmula qué proceso lo representa (su *representación gráfica*). Demostramos en nuestro contexto el resultado análogo al de [BL92]: las cc-fórmulas consistentes y primas se pueden representar por un proceso covariante-contravariante.

Por cuestiones técnicas este estudio se realizó en primer lugar sobre procesos sin acciones bivariantes. Esto se hizo así ya que es necesario que exista un proceso mínimo (con respecto a la cc-simulación) para que su fórmula característica sea $\bot$; pero, si hay acciones bivariantes, no existe un proceso mínimo. Esto responde al bien sabido hecho de que la bisimulación es una equivalencia que no se puede aproximar por una sucesión de órdenes, por lo que las acciones bivariantes (que se rigen por las condiciones de la bisimulación) tampoco pueden aproximarse.

Para tratar de trasladar a los procesos generales con acciones bivariantes los resultados probados, nuestra primera idea fue usar la composición $\mathcal{C} \circ \mathcal{M}$, para "eliminar" las acciones bivariantes del proceso y aplicar los resultados obtenidos. Aunque este acercamiento era perfectamente válido, como ya comentamos antes, la transformación $\mathcal{M}$ introducía un estado extra u lo que hacía que la transformación no resultase demasiado intuitiva. Así, pensamos que si únicamente estábamos interesados en trabajar sin acciones bivariantes, parecía más sensato dividir cada acción $c \in A^{bi}$ en un par de acciones (c^r, c^l), donde c^r era la parte covariante de c y c^l su parte contravariante.

Esta segunda trasformación, $\mathcal{T}$, tiene todas las buenas propiedades de la primera y la gran ventaja de ser mucho más sencilla (siempre que únicamente se quiera transformar

[1] Existen diversas variantes de la definición de subinstitución como se puede encontrar en [GR02], pero todas tienen en común que el funtor usado para tal traslación debe de ser una equivalencia de categorías.

[2] Un objeto Z es débilmente final si para cualquier objeto X existe un morfismo $f : X \to Z$. A diferencia de los objetos finales no se exige que dicho morfismo f sea único. Las equivalencias de categorías preservan los objetos débilmente finales.

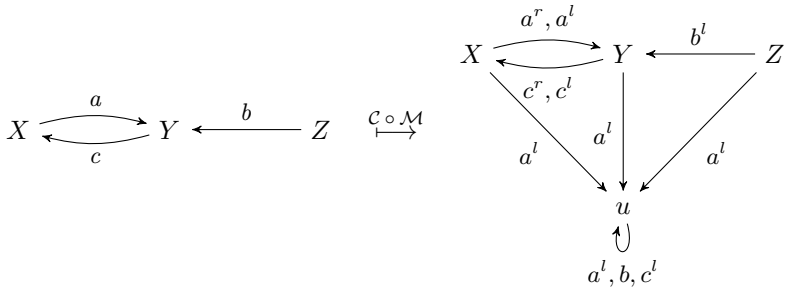


Figura 3.1: La transformación inicial $\mathcal{C} \circ \mathcal{M}$ de S1, donde $A^r = \{a\}$, $A^l = \{b\}$ y $A^{bi} = \{c\}$.

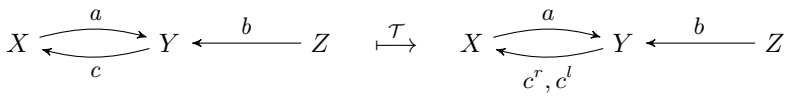


Figura 3.2: La transformación $\mathcal{T}$ de S2, donde $A^r = \{a\}$, $A^l = \{b\}$ y $A^{bi} = \{c\}$.

un LTS con acciones bivariantes en otro sin ellas) que la composición $\mathcal{C} \circ \mathcal{M}$ (véanse las figuras 3.1 y 3.2). Sin embargo, dentro del universo extendido en el que todas las acciones de un LTS tienen una parte covariante y otra contravariante, se puede comprobar que el sistema transformado usando $\mathcal{C} \circ \mathcal{M}$ es el menor posible de entre todos aquellos que reflejan y preservan el orden de la cc-simulación. Esto hace que el uso de dicha transformación inicial resulte claramente interesante si fuésemos a trabajar en dicho marco.

Como comentábamos al inicio de la sección, en realidad, antes de abordar este estudio comparativo entre la simulación covariante-contravariante y los refinamientos modales, habíamos publicado dos contribuciones sobre dos cuestiones básicas para cualquier simulación: su caracterización lógica y la axiomatización del preorden y de la equivalencia inducida. La motivación para este estudio era reproducir el estudio sistemático de las semánticas clásicas que hizo van Glabbeek, aplicado a nuestras dos nuevas semánticas.

En S3 construimos la citada caracterización lógica de las nociones de simulación covariante-contravariante y conforme. Para el caso de la cc-simulación, el punto de partida fue la conocida lógica $\mathcal{L}_S$ que caracteriza a la simulación estándar. Como veíamos en la sección 2.1.1, $\mathcal{L}_S$ contiene tt, la conjunción $\bigwedge_{i \in I} \varphi_i$ y el operador modal existencial $\langle a \rangle \varphi$, definidos de la manera usual. Además, como vimos en el artículo S3, a la lógica $\mathcal{L}_S$ se le pueden añadir tanto la constante ff como la disyunción $\bigvee_{i \in I} \varphi_i$ de manera que se obtiene la lógica $\bar{\mathcal{L}_S}$ que sin embargo sigue caracterizando a la simulación.

Ahora bien, si únicamente tenemos acciones covariantes $a \in A^r$, entonces la cc-simulación coincidiría con la simulación estándar, con lo que la lógica $\mathcal{L}_{CC}$ que caracteriza a la simulación covariante-contravariante debe coincidir con $\bar{\mathcal{L}_S}$. Análogamente, si únicamente hubiese acciones contravariantes $b \in A^l$, entonces $\lesssim_{CC}$ se convierte en la anti-simulación, con lo que la lógica que la caracteriza tendrá que ser la opuesta a $\bar{\mathcal{L}_S}$, es decir, será la lógica compuesta por la negación de las fórmulas en $\bar{\mathcal{L}_S}$. Es más: la negación

de tt es ff, y del operador $\wedge$, el operador $\vee$ (y viceversa); mientras que para el caso del operador $\langle a \rangle$ se obtiene su dual $[a]$, el operador modal universal.

Para obtener la lógica covariante-contravariante basta con observar que los operadores tt, ff, $\wedge$ y $\vee$ pueden usarse con todas las fórmulas, mientras que los operadores modales $\langle a \rangle$ y $[b]$ se deben usar únicamente con las acciones covariantes y contravariantes, respectivamente. Así, obteníamos la siguiente lógica.

> **Definición 3** (extraído de S3.) Dado un alfabeto A, y $\{A^r, A^l, A^{bi}\}$ una partición suya, la clase $\mathcal{L}_{CC}$ de fórmulas covariantes-contravariantes está definida recursivamente por:
>
> - tt y ff están en $\mathcal{L}_{CC}$.
> - Si I es un conjunto y $\varphi_i \in \mathcal{L}_{CC}$ para todo $i \in I$, entonces $\bigwedge_{i\in I} \varphi_i \in \mathcal{L}_{CC}$, $\bigvee_{i\in I} \varphi_i \in \mathcal{L}_{CC}$.
> - Si $\varphi \in \mathcal{L}_{CC}$ y $a \in A^r \cup A^{bi}$ entonces $\langle a \rangle \varphi \in \mathcal{L}_{CC}$.
> - Si $\varphi \in \mathcal{L}_{CC}$ y $a \in A^l \cup A^{bi}$ entonces $[a]\varphi \in \mathcal{L}_{CC}$.
>
> La relación $\models$ se define por:
>
> - $p \models \mathsf{tt}$.
> - $p \models \bigwedge_{i\in I} \varphi_i$ si $p \models \varphi_i$ para todo $i \in I$.
> - $p \models \bigvee_{i\in I} \varphi_i$ si $p \models \varphi_i$ para algún $i \in I$.
> - $p \models \langle a \rangle \varphi$ si existe algún p' tal que $p \xrightarrow{a} p'$ y $p' \models \varphi$.
> - $p \models [a]\varphi$ si $p' \models \varphi$ para todo p' tal que $p \xrightarrow{a} p'$.

Una vez que teníamos una caracterización lógica de las nuevas simulaciones, la otra vía explorada habitualmente para caracterizar las semánticas es la axiomatización de los preórdenes que las definen. Eso fue lo que abordamos en S4, el último artículo del capítulo 5.

Al igual que con la caracterización lógica, el punto de partida para la axiomatización del preorden covariante-contravariante es considerar que en presencia de únicamente las acciones covariantes, la simulación covariante-contravariante debe estar axiomatizada por el axioma $(\mathsf{S}) : x \sqsubseteq x + y$. Análogamente, si únicamente hay acciones contravariantes, entonces la cc-simulación coincide con la anti-simulación que, evidentemente, está axiomatizada por el axioma $(\mathsf{S}^{-1}) : x + y \sqsubseteq x$. Por lo tanto, basta con distinguir el tipo de la acción utilizada como prefijo para así axiomatizar el preorden covariante-contravariante. Es decir, si $a_r \in A^r$ y $a_l \in A^l$, tenemos:

$(\mathsf{S}^r_p) \quad x \sqsubseteq x + a_r y.$

$(\mathsf{S}^l_p) \quad x + a_l y \sqsubseteq x.$

Por otro lado, las acciones bivariantes no necesitan aparecer en los axiomas anteriores ya que se rigen por los axiomas de la bisimulación que veíamos en la sección 2.1.1.

A la hora de axiomatizar las equivalencias inducidas, para el caso de la simulación covariante-contravariante nos encontramos con un sorprendente revés: en presencia de acciones bivariantes la equivalencia inducida no es finitamente axiomatizable. De este modo obteníamos el segundo ejemplo conocido de semántica cuyo preorden puede axiomatizarse finitamente, pero su equivalencia inducida no. Fue también interesante descubrir que Baeten y el resto de autores de [BvBL$^+$10] usaban el mismo conjunto de ecuaciones que nosotros utilizamos para la demostración de la no-axiomatizabilidad, para conjeturar que la equivalencia de bisimulación parcial no era finitamente axiomatizable pues, recordémoslo, la bisimulación parcial es un caso particular de nuestra simulación covariante-contravariante.

Las dificultades encontradas a la hora de axiomatizar la equivalencia inducida para el caso de la semántica covariante-contravariante desaparecieron, exactamente igual que en el caso de la representación gráfica de procesos, una vez que nos restringimos al caso particular que no consideraba acciones bivariantes. La técnica para la axiomatización de este caso particular fue la de considerar la, ya ampliamente usada, caracterización

$$p \equiv p + q \Leftrightarrow q \lesssim p\,.$$

Para el caso particular en el que $A^{bi} = \emptyset$ obtuvimos las siguientes axiomatizaciones:

($\mathsf{S1}^{r,l}_{\equiv}$) $\quad a_r(x + b_r y) = a_r(x + b_r y) + a_r x.$

($\mathsf{S2}^{r,l}_{\equiv}$) $\quad a_r x = a_r x + a_r(x + b_l y).$

($\mathsf{S3}^{r,l}_{\equiv}$) $\quad a_l x = a_l x + a_l(x + b_r y).$

($\mathsf{S4}^{r,l}_{\equiv}$) $\quad a_l(x + b_l y) = a_l(x + b_l y) + a_l x.$

La simulación conforme

La simulación conforme está definida de la siguiente manera:

Definición 3.3.2. *Dados $P = (P, A, \to_P)$ y $Q = (Q, A, \to_Q)$ dos sistemas de transiciones con el alfabeto A, una* simulación conforme *entre ellos es una relación $R \subseteq P \times Q$ tal que siempre que pRq, se tiene:*

- *Para toda $a \in A$, si $p \xrightarrow{a}$, entonces $q \xrightarrow{a}$ (es decir, $I(p) \subseteq I(q)$).*
- *Para toda $a \in A$ tal que $q \xrightarrow{a} q'$ y $p \xrightarrow{a}$, entonces existe p' tal que $p \xrightarrow{a} p'$ y $p'Rq'$.*

Escribiremos $p \lesssim_{CS} q$ si existe una simulación conforme R tal que pRq.

La simulación conforme es también una simulación categórica de Hughes y Jacobs, como vemos a continuación.

> **Proposición 2** (extraído de C1.) La simulación conforme se puede obtener como la simulación coalgebraica para el orden $\sqsubseteq^{Conf}$ sobre el funtor $\mathcal{P}^A$, donde para cada conjunto X tenemos $u \sqsubseteq^{Conf}_X v$ si para cada $u, v : A \to \mathcal{P}X$ y $a \in A$:

- o bien $u(a) = \emptyset$, o
- $u(a) \supseteq v(a)$ con $v(a) \neq \emptyset$.

Como comentábamos antes, la simulación conforme considera que un proceso p es mejor que otro q, siempre que o bien p pueda realizar acciones nuevas que no tenga q (esto lo impone la primera cláusula de la definición), o bien p sea más determinista que q (impuesto por la segunda cláusula). De este modo se tiene que $0 \lesssim_{CS} a0$, pero $ap + aq \lesssim_{CS} ap$. El concepto está ligado a la noción de *conformidad* en *testing*, que se resume en que una implementación es *conforme* a una especificación si esta hace todo lo que la especificación pide de ella, es decir, si ante todo *test* la implementación se comporta como la especificación.

En cierto sentido, también se puede argüir que la simulación covariante-contravariante reduce el no-determinismo, pues desde el punto de vista clásico se considera que las salidas son elegidas por la máquina de manera no-determinista, por lo que un proceso con menos salidas es, en este sentido, más determinista. Pero, por otro lado, también podemos ver a la simulación conforme como una variante de la simulación covariante-contravariante, donde en vez de separar a las acciones en distintos grupos, tenemos un tratamiento común a todas ellas, que diferencia dos casos: si esa acción ya estaba en el proceso, o si no lo estaba. De hecho, la segunda cláusula de la simulación conforme es puramente contravariante, con lo que este hecho de reducir el no-determinismo es común a ambas.

Como también tratábamos en S3, para construir la lógica que caracteriza a la simulación conforme, el punto de partida fue la propia lógica covariante-contravariante que veíamos en la página 68. Como acabamos de observar, la simulación conforme tiene una segunda cláusula contravariante, lo que nos hizo pensar que necesitaríamos un operador modal con un comportamiento universal. Pero ya que también se tiene $0 \lesssim_{CS} a$, a su vez el operador debía capturar este sentido existencial. La respuesta fue combinar los dos operadores modales de la lógica $\mathcal{L}_{CC}$ para obtener un nuevo operador modal a, cuya semántica es: $p \models a\varphi$ *si* $p \xrightarrow{a} y\ p' \models \varphi$ *para todo* $p \xrightarrow{a} p'$. De hecho, se tiene que $a\varphi$ es equivalente a $\langle a \rangle \varphi \wedge [a]\varphi$. Y con ello obtuvimos la siguiente definición.

Definición 4 (extraído de S3.) La clase de fórmulas conformes $\mathcal{L}_{CS}$ se define recursivamente por:

- $\mathsf{tt} \in \mathcal{L}_{CS}$.
- Si I es un conjunto y $\varphi_i \in \mathcal{L}_{CC}$ para todo $i \in I$, entonces $\bigwedge_{i \in I} \varphi_i \in \mathcal{L}_{CC}$, $\bigvee_{i \in I} \varphi_i \in \mathcal{L}_{CC}$.
- Si $\varphi \in \mathcal{L}_{CS}$ y $a \in A$, entonces $a\varphi \in \mathcal{L}_{CS}$.

La relación $\models$ está definida por:

- $p \models \mathsf{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ si $p \models \varphi_i$ para todo $i \in I$.
- $p \models \bigvee_{i \in I} \varphi_i$ si $p \models \varphi_i$ para algún $i \in I$.
- $p \models a\varphi$ si $p \xrightarrow{a}$ y $p' \models \varphi$ para todo $p \xrightarrow{a} p'$.

Al igual que en el caso de la simulación covariante-contravariante, en la publicación S4, estudiábamos la caracterización axiomática del preorden conforme. Este caso era un poco más complejo que el de la simulación covariante-contravariante pues, para empezar, la simulación no es una precongruencia con respecto a la suma de procesos: $0 \lesssim_{CS} ab$, $ac \lesssim_{CS} ac$, pero no se tiene $ac \lesssim_{CS} ab + ac$.

La solución que escogimos fue la misma que se tomó en [Mil89] para la bisimulación débil, y es la de trabajar con la precongruencia más débil contenida en la simulación conforme:

$$p \lesssim^{p}_{CS} q \text{ si y solo si } p \lesssim_{CS} q \text{ e } I(p) \supseteq I(q)\,.$$

Lo más importante es observar que no es necesario imponer la condición $I(p) \supseteq I(q)$ de manera recursiva, sino únicamente en la raíz del proceso; además, tampoco es necesario imponer explícitamente $I(p) \subseteq I(q)$, pues esto siempre se tiene por la definición de simulación conforme.

La precongruencia conforme es finitamente axiomatizable y basta considerar los dos casos que la definen: añadir acciones nuevas al proceso es siempre positivo, mientras que añadir acciones que ya estuviesen se considera perjudicial. Es decir, obtuvimos los siguientes axiomas:

$(\mathsf{S}_{CS,g})$ $\quad I(p) \cap I(q) = \emptyset \Longrightarrow ap \sqsubseteq a(p+q).$

$(\mathsf{S}^{-1}_{CS,p})$ $\quad ap + aq \sqsubseteq ap.$

De nuevo, a la hora de axiomatizar la equivalencia inducida usamos la caracterización $p \equiv p + q \Leftrightarrow q \lesssim p$. Este caso no entrañó otras dificultades que la propia complejidad de las demostraciones y tampoco hubo necesidad de definir una precongruencia, ya que la equivalencia inducida ya lo era. Así, se obtuvieron los siguientes axiomas:

$(\mathsf{S}^{CS}_{\equiv})$ $\quad I(p) \cap I(q) = \emptyset \Longrightarrow ap = ap + a(p+q).$

$(\mathsf{S}^{-1,CS}_{\equiv})$ $\quad I(q) \subseteq I(p) \Longrightarrow a(p+q) = a(p+q) + ap.$

Lo más interesante de esta axiomatización fue que nos permitió percatarnos de que el axioma $(\mathsf{S}^{-1}_{CS,p})$ es el que caracteriza la equivalencia de *ready simulation*. De este modo, la noción de *ready conformance simulation* coincide con el inverso de la *ready simulation*, lo que viene a decirnos que todas aquellas semánticas que están por debajo de esta coinciden (pero con el orden revertido) con las nociones conformes correspondientes. Así se obtiene, por ejemplo, una noción mucho más intuitiva de *fallos conformes*, que considera que el proceso que tiene más fallos es peor que el que tiene menos.

De esta manera, concluíamos (por ahora) el objetivo 2.2. Creemos que con lo logrado se ilustra suficientemente el interés de estas dos nociones de simulación. La simulación covariante-contravariante es un sencillo pero potente ejercicio de unificación de las quizá dos nociones más importantes de relación entre procesos: la simulación y la bisimulación.

Por su parte, la simulación conforme es en nuestra opinión una noción de simulación muchísimo más intuitiva que la clásica en el sentido de que no podemos "mejorar" (simular) un proceso haciéndolo más no-determinista. Esta "debilidad" de los órdenes clásicos

de simulación se transmite a las semánticas lineales asociadas: la de trazas y la de fallos, en las que de nuevo los procesos menos deterministas pueden mayorar a otros más deterministas. Por contra, la semántica de trazas conformes correspondiente a la simulación conforme, "arregla" satisfactoriamente ese problema.

Capítulo 4

Conclusiones y trabajo futuro

Comentábamos en la sección 1.4.1 que la meta general de nuestra investigación era la de comprender mejor la estrecha relación existente entre las semánticas de procesos y el mundo categórico de las coálgebras. Entendimos que el punto de partida más razonable para lograrlo era la, por aquel entonces, reciente definición de simulación coalgebraica dada por Hughes y Jacobs.

Como hemos visto en las publicaciones que hemos recogido en el capítulo anterior, a nuestro entender hemos conseguido importantes avances en el estudio de la simulación coalgebraica. Hemos estudiado la refexión y preservación de propiedades lógicas por las mismas; cómo las transformaciones naturales mantienen las simulaciones; y los diversos pormenores técnicos de la condición de estabilidad. A lo largo de estos artículos hemos sido capaces de trasladar al marco general categórico un buen número de resultados de la simulación clásica, y de encontrar aplicaciones concretas en relación con las semánticas de procesos clásicas y nuevas variantes de las mismas. En concreto, hemos utilizado los sistemas de multitransiciones como un marco unificador para los sistemas de transiciones y los sistemas de transiciones probabilísticos, y hemos obtenido dos nuevas semánticas de simulación, de las que hemos justificado su utilidad, interés y buenas propiedades.

Por el modo en que ha sido concebida, todos los resultados de la presente tesis vienen ya avalados por su publicación previa en las actas de diversos congresos internacionales de reconocido prestigio. Ello supone de hecho un doble aval, pues los exigentes procesos de evaluación y selección en competencia directa con otros trabajos, aportan además detallados informes de los evaluadores, que sin duda han contribuido a mejorar la calidad de la presentación, incluyendo la corrección de posibles errores u omisiones en las versiones preliminares. Además, nuestro trabajo es ya desconocido por parte de la comunidad científica especializada en el área, que también ha tenido ocasión de opinar y hacer sugerencias en relación al mismo, que también ha contribuido a mejorar la calidad del trabajo, y que por supuesto agradecemos muy sinceramente. De hecho contamos ya con la referencia [AILS11] donde se construyen las fórmulas características para nuestra simulación conforme, utilizándose por tanto la noción que en los trabajos de esta tesis hemos introducido, y mostrando de esta forma el interés que ha suscitado.

Pero ciertamente la investigación en este campo no termina con esta tesis, y la vía

de estudio que hemos abierto aquí tiene diversas posibles continuaciones. Para empezar, podemos comentar un par de direcciones, que más que trabajo futuro, deberíamos calificar como "trabajo actual" pues confiamos en poder obtener nuevos resultados que puedan ser presentados en próximas publicaciones.

Ya comentamos en la sección 2.2.3 que existía otro modo de definir las simulaciones desde el punto de vista coalgebraico, en este caso, por medio de *relatores*. Además, como decíamos en la sección 2.2.4, Corina Cîrstea había definido un método general y categórico para obtener la lógica que caracteriza a una simulación. De este modo podemos estudiar las lógicas modales que caracterizan distintas nociones de simulación desde una perspectiva puramente categórica. En este próximo primer trabajo esperamos poder definir como relatores tanto nuestras dos semánticas de simulación (la cc-simulacíón y la simulación conforme), como las nociones de refinamiento modal, la simulación entre sistemas de transiciones mixtos y la bisimulación parcial. Una vez que hayamos trasladado estos conceptos al marco de los relatores, pretendemos reproducir el esquema de trabajo de [Cîr06] para obtener las lógicas que las caracterizan.

Este proyecto tiene un doble interés. En primer lugar pretendemos mostrar y explicar cómo la teoría desarrollada en [Cîr06] puede aplicarse a ejemplos no estándar, como son nuestras dos simulaciones. En segundo lugar, por lo que sabemos, aún no se ha trabajado en el modelo coalgebraico con los sistemas modales y mixtos. Más en concreto: no existen referencias de que se tenga definida una lógica que caracterice el preorden de la simulación entre MiTS. Pues bien, confiamos en comenzar a ir rellenando esta laguna en un futuro próximo.

Nuestro segundo trabajo inmediato tiene como marco la extensión del espectro *linear time-branching time* de van Glabbeek a nuestras semánticas covariante-contravariante y conforme. Esta idea surgió cuando estábamos estudiando las axiomatizaciones de los preórdenes en el artículo S2. De hecho, ya comentábamos en esa publicación que para la simulación conforme la noción de *ready conformance simulation* coincidía con el opuesto de la *ready simulation*. Nos pareció entonces oportuno tratar de extender este resultado obteniendo las nociones adecuadas de trazas o fallos, entre otras, para estos contextos.

Proponemos empezar entonces con las definiciones adecuadas de *trazas covariantes-contravariantes* y *trazas conformes*. Para el caso de las trazas covariantes-contravariantes, como esperamos no resulte sorprendente tras estudiar las técnicas que hemos utilizado en la tesis, la idea es componer primero la noción de traza ordinaria (que correspondería al caso covariante) con la noción de *no-trazas* (para el caso contravariante). Éstas últimas no son más que aquellas secuencias de acciones que un proceso no puede ejecutar. Combinando ambas nociones, obtenemos las trazas covariantes-contravariantes.

En el caso de las *trazas conformes* aparece el concepto de *traza segura* que corresponde a aquellas trazas que el proceso siempre puede ejecutar, como contraposición a las trazas usuales, en las que, debido al no-determinismo, podemos encontrarnos con situaciones en las que cuando el proceso trata de ejecutar una traza suya se encuentra con un camino en el que no es capaz de culminar su ejecución.

Partiendo de las trazas generalizadas continuaremos con el resto de las semánticas más conocidas: *ready simulation*, fallos y *readiness*. De nuevo, resultará especialmente útil estudiar, en primer lugar, cómo estas semánticas se comportan en el caso puramente contravariante, para después discutir la manera natural de "mezclarlo" con el caso en el

que también hay acciones covariantes, para obtener las definiciones generales para el caso covariante-contravariante.

Al margen de estas dos direcciones que estamos ya explorando en profundidad en la actualidad, quedan otras vías de desarrollo que merece la pena comentar. Un tema que barajamos introducir en nuestro estudio durante los inicios de la tesis fueron las diversas nociones de simulación que aparecen en [Pal05]. De entre ellas, nos parece especialmente interesante la noción de *simulación tartamuda*, definida por Manolios en [Man01], que generaliza la noción de simulación estándar al permitir que dos procesos sean similares si después de que un proceso haga una serie de transiciones (un *camino* en el contexto), el otro pueda hacer otro camino de manera que ambos "encajen". Formalmente, dados dos sistemas de transiciones y una relación H, un camino π *H-encaja* con otro ρ, si existen dos funciones estrictamente crecientes $\alpha, \beta : \mathbb{N} \to \mathbb{N}$ con $\alpha(0) = \beta(0) = 0$ tales que para todo $i, j, k \in \mathbb{N}$ si $\alpha(i) \leq j < \alpha(i+1)$ y $\beta(i) \leq k < \beta(i+1)$, se cumple que $\alpha(j) H \beta(k)$. En consecuencia, la simulación tartamuda es una relación binaria H tal que si xHy, entonces para cada camino π que comience en x existe un camino ρ que comienza en y y que encaja con π.

También sería interesante estudiar las semánticas débiles de bisimulación y simulación. En concreto, en los trabajos de Sokolova, Vink y Woracek [SVW04, SdVW09], se define ya la bisimulación débil desde el punto de vista coalgebraico. Para ello se comienza extendiendo el alfabeto de acciones con las denominadas acciones internas τ. Se debe entonces extender el comportamiento de las coálgebras para que actúen sobre palabras en vez de sobre acciones y, finalmente, se abstraen las acciones internas. Un primer trabajo interesante nos llevaría a estudiar si estas construcciones se mantienen una vez que se usen los órdenes funtoriales.

En relación con las simulaciones covariante-contravariante y conforme, otra prometedora vía de estudio es seguir aprovechando las transformaciones que comentábamos en la página 66, y que definíamos en los artículos S1 y S2. En concreto, en S1 definíamos dos transformaciones: $\mathcal{M}$ y $\mathcal{C}$ entre MTS y LTS, que nos permitían convertir un LTS en un MTS, y viceversa. En la publicación S4 caracterizamos axiomáticamente tanto el preorden como la equivalencia inducida por la simulación covariante-contravariante. Ahora nos proponemos estudiar de qué manera y en qué condiciones es posible trasladar las citadas axiomatizaciones a los MTS. De esta manera podríamos caracterizar axiomáticamente el refinamiento entre MTS.

Por otro lado, la transformación $\mathcal{T}$ que definíamos en S2 nos permitía considerar sistemas covariantes-contravariantes en los que no aparecen acciones bivariantes. Como también comentábamos en S4, la equivalencia inducida por la cc-simulación no es finitamente axiomatizable en el caso de que aparezcan acciones bivariantes mezcladas con otras acciones covariantes o contravariantes. La transformación $\mathcal{T}$, al permitirnos desprendernos de las acciones bivariantes, puede servirnos para estudiar la axiomatización de la equivalencia de cc-simulación en un contexto en el que las acciones bivariantes a se ven sustituidas por el par asociado (a^r, a^l).

Siguiendo esta línea referente a la axiomatización de nuestras nociones de simulación, un siguiente paso es el de incorporar a BCCSP nuevos operadores, comenzando por el operador de composición *paralela*, y estudiar de qué manera han de ser extendidas nuestras caracterizaciones axiomáticas y lógicas, para cubrir las nuevas operaciones.

Dejamos para el final una meta que consideramos mucho más especulativa que las anteriormente comentadas: estudiar la posible relación de la formalización de la semántica con coálgebras con los operadores clásicos de las semánticas de procesos. En particular, tanto el operador de composición paralela como el de composición secuencial pueden ser un magnífico punto de partida, que nos permitirían tener nociones categóricas de cómo dos coálgebras cualesquiera pueden componerse en ciertas formas "naturales". Ciertamente, de conseguirse avances sustanciales en esta dirección estaríamos estrechando mucho más aún la relación entre las coálgebras y las semánticas de procesos.

Bibliografía

[Acz88] Peter Aczel. *Non-Well-Founded Sets*, volumen 14. CSLI Lecture Notes, 1988.

[AFV01] Luca Aceto, Willem Jan Fokkink, y Chris Verhoef. Structural operational semantics. En Jan A. Bergstra, Alban Ponse, y Scott A. Smolka, editores, *Handbook of Process Algebra*, páginas 197–292. Elsevier, 2001.

[AHK97a] Rajeev Alur, Thomas A. Henzinger, y Orna Kupferman. Alternating-time temporal logic. En Willem P. de Roever, Hans Langmaack, y Amir Pnueli, editores, *COMPOS*, volumen 1536 de *Lecture Notes in Computer Science*, páginas 23–60. Springer, 1997.

[AHK97b] Rajeev Alur, Thomas A. Henzinger, y Orna Kupferman. Alternating-time temporal logic. En *FOCS*, páginas 100–109, 1997.

[AHKV98] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, y Moshe Y. Vardi. Alternating refinement relations. En Davide Sangiorgi y Robert de Simone, editores, *CONCUR*, volumen 1466 de *Lecture Notes in Computer Science*, páginas 163–178. Springer, 1998.

[AHL+08] Adam Antonik, Michael Huth, Kim Larsen, Ulrik Nyman, y Andrzej Wasowski. 20 Years of Mixed and Modal Specifications. *Bulletin of the European Association for Theoretical Computer Science*, May 2008.

[AILS11] Luca Aceto, Anna Ingolfsdottir, Paul Blain Levy, y Joshua Sack. Characteristic formulae for fixed-point semantics: A general framework. *Mathematical Structures in Computer Science, special issue devoted to selected papers from EXPRESS 2009*, 2011.

[AM80] M. A. Arbib y E. G. Manes. Machines in a category. *Journal of Pure & Applied Algebra*, 19:9–20, 1980.

[AM89] Peter Aczel y Nax Paul Mendler. A final coalgebra theorem. En David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, y Axel Poigné, editores, *Category Theory and Computer Science*, volumen 389 de *Lecture Notes in Computer Science*, páginas 357–365. Springer, 1989.

[Bae05] Jos C. M. Baeten. A brief history of process algebra. *Theor. Comput. Sci.*, 335(2-3):131–146, 2005.

[Bal00] Alexandru Baltag. A logic for coalgebraic simulation. *Electr. Notes Theor. Comput. Sci.*, 33:42–60, 2000.

[BL92] Gérard Boudol y Kim Gulstrand Larsen. Graphical versus logical specifications. *Theoretical Computer Science*, 106(1):3–20, 1992.

[BvBL+10] J.C.M. Baeten, D.A. van Beek, B. Luttik, J. Markovski, y J.E. Rooda. Partial bisimulation. Technical Report SE 10/04, Eindhoven University of Technology, 2010.

[BW99] Michael Barr y Charles Wells. *Category Theory for Computing Science. Third Edition.* Centre de Recherches Mathématiques, 1999.

[Cîr06] Corina Cîrstea. A modular approach to defining and characterising notions of simulation. *Information and Computation*, 204(4):469–502, 2006.

[Car] Rudolph Carnap. *Logische Syntax der Sprache.* Schriften zur wissenschaftlichen Weltauffassung, Bd. 8. J. Springer.

[CGL93] Karlis Cerans, Jens Chr. Godskesen, y Kim Guldstrand Larsen. Timed modal specification - theory and tools. En Costas Courcoubetis, editor, *CAV*, volumen 697 de *Lecture Notes in Computer Science*, páginas 253–267. Springer, 1993.

[CGP99] Edmund M. Clarke, Orna Grumberg, y Doron A. Peled. *Model Checking.* MIT Press, 1999.

[CLSV04] Ling Cheung, Nancy A. Lynch, Roberto Segala, y Frits W. Vaandrager. Switched probabilistic i/o automata. En Zhiming Liu y Keijiro Araki, editores, *ICTAC*, volumen 3407 de *Lecture Notes in Computer Science*, páginas 494–510. Springer, 2004.

[dAH01] Luca de Alfaro y Thomas A. Henzinger. Interface automata. En *ESEC / SIGSOFT FSE*, páginas 109–120, 2001.

[Dam96] Dennis Dams. *Abstract Interpretation and Partition Refinement for Model Checking.* Tesis Doctoral, Eindhoven University of Technology, july 1996.

[DGG97] Dennis Dams, Rob Gerth, y Orna Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems*, 19:253–291, 1997.

[Dij68] Edsger W. Dijkstra. Cooperating sequential processes. páginas 43–112, 1968.

[dVR99] Erik P. de Vink y Jan J. M. M. Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. *Theoretical Computer Science*, 221(1-2):271–293, 1999.

[EM42] Samuel Eilenberg y Saunders MacLane. Group extensions and homology. *Annals of Mathematics*, 43:757–831, 1942.

[EM45] Samuel Eilenberg y Saunders MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58:231–294, 1945.

[Fáb06] Ignacio Fábregas. Un estudio coalgebraico de las simulaciones. Master's thesis, Facultad de Matemáticas, Universidad Complutense de Madrid, 2006.

[FS08] Harald Fecher y Heiko Schmidt. Comparing disjunctive modal transition systems with an one-selecting variant. *Journal of Logic and Algebraic Programming*, 77(1-2):20–39, 2008.

[GB92] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[GR02] Joseph Goguen and Grigore Roşu. Institution morphisms. *Formal Aspects of Computing*, 13(3-5):274–307, 2002.

[Gro57] Alexandre Grothendieck. Sur quelques points d'algèbre homologique. *Tohoku Mathematics Journal*, 9:119–221, 1957.

[GTWW77] Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. Initial algebra semantics and continuous algebras. *J. ACM*, 24(1):68–95, 1977.

[Hen88] Matthew Hennessy. *Algebraic theory of processes*. MIT Press, 1988.

[Her93] Claudio Hermida. *Fibrations, Logical Predicates and Indeterminates*. Tesis Doctoral, University of Edinburgh, 1993.

[HF89] Joseph Y. Halpern y Ronald Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–177, 1989.

[HJ98] Claudio Hermida y Bart Jacobs. Structural induction and coinduction in a fibrational setting. *Information and Computation*, 145(2):107–152, 1998.

[HJ04] Jesse Hughes y Bart Jacobs. Simulations in coalgebra. *Theoretical Computer Science*, 327(1-2):71–108, 2004.

[HM85] Matthew Hennessy y Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Jac] Bart Jacobs. Introduction to coalgebra. Towards mathematics of states and observations. Libro en preparación. Disponible en `http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`.

[Jac99] Bart Jacobs. *Categorical Logic and Type Theory*, volumen 141 de *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1999.

[JH03] Bart Jacobs y Jesse Hughes. Simulations in coalgebra. *Electr. Notes Theor. Comput. Sci.*, 82(1):128–149, 2003.

[JR97] Bart Jacobs y Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 62:222–259, 1997.

[Kan58] Daniel Kan. Adjoint funtors. *Transactions of the American Mathematical Society*, 87:294–329, 1958.

[KFA69] R. E. Kalman, P. L. Falb, y M. A. Arbib. Topics in mathematical system theory. *Series in Pure & Applied Mathematics*, 1969.

[KLSV03] Dilsun Kirli Kaynar, Nancy A. Lynch, Roberto Segala, y Frits W. Vaandrager. Timed i/o automata: A mathematical framework for modeling and analyzing real-time systems. En *RTSS*, páginas 166–177. IEEE Computer Society, 2003.

[KPH61] T. Kilburn, R. B. Payne, y D. J. Howarth. The atlas supervisor. *AFIPS Computer Conference*, 20:279–294, 1961.

[Kur98] A. Kurz. Coalgebras and modal logic. En *In Proceedings of Advances in Modal Logic '98, Uppsala*, páginas 222–230, 1998.

[Kur00] Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science*. Tesis Doctoral, Universität München, 2000.

[Lam68] Joachim Lambek. A fixpoint theorem for complete categories. *Mathematische Zeitschrift*, 103(2):151–161, 1968.

[Lan66] Peter J. Landin. A lambda-calculus approach. En L. Fox, editor, *Advances in Programming and Nonnumerical Computation*, chapter 5, páginas 97–157. Pergamon Press, 1966.

[Lin04] *One hundred years of Russell's paradox*. Godehard Link, 2004.

[LNW07] Kim Guldstrand Larsen, Ulrik Nyman, y Andrzej Wasowski. Modal i/o automata for interface and product line theories. En Rocco De Nicola, editor, *ESOP*, volumen 4421 de *Lecture Notes in Computer Science*, páginas 64–79. Springer, 2007.

[LS91] Kim Guldstrand Larsen y Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[LT88] Kim Guldstrand Larsen y Bent Thomsen. A modal process logic. En *LICS*, páginas 203–210. IEEE Computer Society, 1988.

[LX90] Kim Guldstrand Larsen y Liu Xinxin. Equation solving using modal transition systems. En *LICS*, páginas 108–117. IEEE Computer Society, 1990.

[Lyn88] Nancy Lynch. I/O automata: A model for discrete event systems. *En 22nd Annual Conference on Information Sciences and Systems*, páginas 29–38, 1988.

[Mac98] Saunders Mac Lane. *Categories for the Working Mathematician. Second Edition.* Springer, 1998.

[Man01] Panagiotis Manolios. *Mechanical Verification of Reactive Systems.* Tesis Doctoral, University of Texas at Austin, August 2001.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*, volumen 92 de *Lecture Notes in Computer Science.* Springer, 1980.

[Mil82] Robin Milner. *A Calculus of Communicating Systems.* Springer, 1982.

[Mil89] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mos99] Lawrence S. Moss. Coalgebraic logic. *Ann. Pure Appl. Logic*, 96(1-3):277–317, 1999.

[MP92] Zohar Manna y Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems. Specification.* Springer, 1992.

[Oll76] A. Ollengren. Definition of programming languages by interpreting automata. 1976.

[Pal05] Miguel Palomino. *Reflexión, abstracción y simulación en la lógica de reescritura.* Tesis Doctoral, Universidad Complutense de Madrid, Spain, March 2005.

[Par81] David Park. Concurrency and automata on infinite sequences. En Peter Deussen, editor, *Theoretical Computer Science, 5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981, Proceedings*, volumen 104 de *Lecture Notes in Computer Science*, páginas 167–183. Springer, 1981.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aahus University, 1981.

[Plo04] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

[Pnu77] A. Pnueli. The temporal logic of programs. En *Proceedings of the* 18^{th} *IEEE Symposium on Foundations of Computer Science*, páginas 46–57. IEEE Computer Society Press, 1977.

[RBB+09] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, y Roberto Passerone. Modal interfaces: unifying interface automata and modal specifications. En *EMSOFT '09: Proceedings of the seventh ACM international conference on Embedded software*, páginas 87–96, New York, NY, USA, 2009. ACM.

[Röß00] Martin Rößiger. Coalgebras and modal logic. *Electr. Notes Theor. Comput. Sci.*, 33:294–315, 2000.

[RT92] Jan J. M. M. Rutten y Daniele Turi. On the foundation of final semantics: Non-standard sets, metric spaces, partial orders. En J. W. de Bakker, Willem P. de Roever, y Grzegorz Rozenberg, editores, *REX Workshop*, volumen 666 de *Lecture Notes in Computer Science*, páginas 477–530. Springer, 1992.

[RT93] Jan J. M. M. Rutten y Daniele Turi. Initial algebra and final coalgebra semantics for concurrency. En J. W. de Bakker, Willem P. de Roever, y Grzegorz Rozenberg, editores, *REX School/Symposium*, volumen 803 de *Lecture Notes in Computer Science*, páginas 530–582. Springer, 1993.

[Rut00] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[Rut01] Jan J. M. M. Rutten. Elements of stream calculus (an extensive exercise in coinduction). *Electr. Notes Theor. Comput. Sci.*, 45:358–423, 2001.

[San07] Davide Sangiorgi. On the origins of Bisimulation, Coinduction and Fixed Points. Technical Report 24, Department of Computer Science, University of Bologna, 2007.

[Sch06a] Heiko A. Schmidt. Comparing disjuntive modal transition systems with an one-selecting variant. En *NWPT'06 - The 18th Nordic Workshop on Programming Theory, Reykjavík, Iceland. Proceedings.* Reykjavík University, 2006.

[Sch06b] Heiko A. Schmidt. Comparing disjuntive modal transition systems with their one-selecting variant. Master's thesis, Christian-Albrechts-Universität zu Kiel, 2006.

[SdVW09] Ana Sokolova, Erik P. de Vink, y Harald Woracek. Coalgebraic weak bisimulation for action-type systems. *Sci. Ann. Comp. Sci.*, 19:93–144, 2009.

[Seg68] Krister Segerberg. Decidability of S4.1. *Theoria*, 34:7–20, 1968.

[Seg71] Krister Segerberg. An essay in classical modal logic. *Filosofiska Studier*, 1971.

[Sha53] Lloyd S. Shapley. Stochastic games. En *NationalAcademy of Science*, volumen 39, páginas 1095–1100, 1953.

[SVW04] A. Sokolova, E.P. de Vink, y H. Woracek. Weak bisimulation for action-type coalgebras. Technical Report CSR 04–16, Technische Universiteit Eindhoven, 2004.

[Thi96] Albert Thijs. *Simulation and fixpoint semantics.* Tesis Doctoral, Rijksuniversiteit Groningen, 1996.

[TR98] Daniele Turi y Jan J. M. M. Rutten. On the foundations of final coalgebra semantics. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998.

[Tur36] Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[vB76] Johan van Benthem. *Modal Correspondence Theory.* Tesis Doctoral, Mathematish Instituut & Instituut voor Grondslagenonderzoek, University pf Amsterdam, 1976.

[vG01] R. J. van Glabbeek. The linear time-branching time spectrum I: The semantics of concrete, sequential processes. En J. A. Bergstra, A. Ponse, y S. A. Smolka, editores, *Handbook of process algebra*, páginas 3–99. North-Holland, 2001.

[Weg72] P. Wegner. The Vienna definition language. *ACM Computing Surveys*, 4(1):5–63, 1972.

Capítulo 5

Publicaciones

En este capítulo incluimos las siete publicaciones que componen la tesis. Como ya hicimos a la hora de exponer los resultados en el capítulo 3, en lugar de presentar los artículos en orden cronológico lo hacemos de modo temático, destacando el diferente carácter de los mismos.

En el primer bloque aparecen las tres publicaciones de corte más categórico. La primera fue aceptada en 2007 en el congreso *International Colloquium on Theoretical Aspects of Computing* (ICTAC), en ella tratamos la reflexión y preservación de propiedades lógicas por parte de las (bi)simulaciones coalgebraicas.

La segunda de las publicaciones apareció en 2009, en el congreso *Conference on Algebra and Coalgebra in Computer Science* (CALCO). En ella discutimos los resultados de estabilidad de las simulaciones y definimos por primera vez las simulaciones *covariante-contravariante* y *conforme*.

El último artículo del bloque categórico se publicó en las actas de 2008 del congreso *International Conference on Formal Techniques for Networked and Distributed Systems* (FORTE). En él presentamos los sistemas de multitransiciones como marco unificador y discutimos la relación entre las transformaciones naturales y las (bi)simulaciones coalgebraicas.

El segundo bloque incluye las cuatro publicaciones en las que estudiamos y relacionamos nuestras dos semánticas de simulación. La primera se aceptó para el congreso *International Conference on Fundamentals of Software Engineering* (FSEN) en este año 2011, y la dedicamos a relacionar la simulación covariante-contravariante con la bisimulación parcial y los refinamientos modales. Aún no ha aparecido publicada en el volumen 7141 de *Lecture Notes in Theoretical Computer Science*.

Además, los miembros del comité de programa del congreso FSEN'11 han preseleccionado a esta publicación (junto con otras 7) para que los autores preparemos una versión extendida para su posible publicación en un número especial de la revista "Science of Computer Programming" (SCP).

La segunda ha sido aceptada para el congreso *Expressiveness in Concurrency* (EXPRESS) también este año 2011. En esta publicación reproducimos el estudio de Boudol y Larsen con nuestra simulación covariante-contravariante. Como en el caso anterior, los

miembros del congreso han preseleccionado este artículo invitándonos a los autores a preparar una versión extendida para su posible publicación en un número especial de la revista "Mathematical Structures in Computer Science" (MSCS).

La tercera publicación es de 2010 y apareció en el congreso *International Conference on Formal Techniques for Distributed Systems* (FMOODS-FORTE). Es la publicación en la que construimos las lógicas para las dos semánticas de simulación.

Finalmente, el artículo que cierra el capítulo se presentó en 2010 en *Structural Operational Semantics* (SOS) y lo dedicamos al estudio de las axiomatizaciones de nuestras semánticas.

Reflection and Preservation of Properties in Coalgebraic (bi)Simulations*

Ignacio Fábregas, Miguel Palomino, and David de Frutos Escrig

Departamento de Sistemas Informáticos y Computación, UCM
fabregas@fdi.ucm.es, {miguelpt, defrutos}@sip.ucm.es

Abstract. Our objective is to extend the standard results of preservation and reflection of properties by bisimulations to the coalgebraic setting, as well as to study under what conditions these results hold for simulations. The notion of bisimulation is the classical one, while for simulations we use that proposed by Hughes and Jacobs. As for properties, we start by using a generalization of linear temporal logic to arbitrary coalgebras suggested by Jacobs, and then an extension by Kurtz which includes atomic propositions too.

1 Introduction

To reason about computational systems it is customary to mathematically formalize them by means of state-based structures such as labelled transitions systems or Kripke structures. This is a fruitful approach since it allows to study the properties of a system by relating it to some other, possibly better-known system, by means of simulations and bisimulations (see e.g., [15,14,12,3]).

The range of structures used to formalize computational systems is quite wide. In this context, coalgebras have emerged with a unifying aim [18]. A coalgebra is simply a function $c : X \longrightarrow FX$, where X is the set of states and FX is some expression on X (a functor) that describes the possible outcomes of a transition from a given state. Choosing different expressions for F one can obtain coalgebras that correspond to transition systems, Kripke structures, ...

Coalgebras can also be related by means of (bi)simulations. Our goal in this paper is to prove that, like their concrete instantiations, (bi)simulations between arbitrary coalgebras preserve some interesting properties. A first step in this direction consists in choosing an appropriate notion for both bisimulation and simulation, as well as a logic in which to express these properties.

Bisimulations were originally introduced by Aczel and Mendler [1], who showed that the general definition coincided with the standard ones when particularized; it is an established notion. Simulations, on the other hand, were defined by Hughes and Jacobs [8] and lack such canonicity. Their notion of simulation depends on the use of orders that allow (perhaps too) much flexibility in what it can be considered as a simulation; in order to show that simulations preserve properties, we

* Research supported by the Spanish projects DESAFIOS TIN2006-15660-C02-01, WEST TIN2006-15578-C02-01 and PROMESAS S-0505/TIC/0407.

C.B. Jones, Z. Liu, J. Woodcock (Eds.): ICTAC 2007, LNCS 4711, pp. 231–245, 2007.

will have to impose certain restrictions on such orders. As for the logic used for the properties, there is likewise no canonical choice at the moment. Jacobs proposes a temporal logic (see [9]) that generalizes linear temporal logic (LTL), though without atomic propositions; a clever insight of Pattinson [17] provides us with a way to endow Jacobs' logic with atomic propositions.

Since our original motivation was the generalization of the results about simulations and preservation of LTL properties, we will focus on Jacobs' logic and its extension with atomic propositions. Actually, modal logic seems to be the right logic to express properties of coalgebras and several proposals have been made in this direction, among them those in [10,13,17], which are invariant under behavioral equivalence. The reason for studying preservation/reflection of properties by bisimulations here is twofold: on the one hand, some of the operators in Jacobs' logic do not seem to fall under the framework of those general proposals; on the other hand, some of the ideas and insights developed for that study are needed when tackling simulations. As far as we know, reflection of properties by simulations in coalgebras has not been considered before in the literature.

2 Preliminaries

In this section we summarize definitions and concepts from [8,11,9], and introduce the notation we are going to use.

Given a category $\mathbb{C}$ and an endofunctor F in $\mathbb{C}$, an F-coalgebra, or just a coalgebra, consists of an object $X \in \mathbb{C}$ together with a morphism $c : X \longrightarrow FX$. We often call X the state space and c the transition or coalgebra structure.

Example 1. We show how two well-known structures can be seen as coalgebras:

- Labelled transition systems are coalgebras for the functor $F = \mathcal{P}(id)^A$, where A is the set of labels.
- Kripke structures are coalgebras for the functor $F = \mathcal{P}(AP) \times \mathcal{P}(id)$, where AP is a set of atomic propositions.

It is well-known that an arbitrary endofunctor F on **Sets** can be lifted to a functor in the category **Rel** of relations, that is, $\mathrm{Rel}(F) : \mathbf{Rel} \longrightarrow \mathbf{Rel}$. Given a relation $R \subseteq X \times Y$, its lifting is defined by

$$\mathrm{Rel}(F)(R) = \{\langle u, v\rangle \in FX_1 \times FX_2 \mid \exists w \in F(R).\, F(r_1)(w) = u, F(r_2)(w) = v\}\ ,$$

where $r_i : R \longrightarrow X_i$ are the projection morphisms.

A predicate P of a coalgebra $c : X \longrightarrow FX$ is just a subset of the state space. Also, a predicate $P \subseteq X$ can be lifted to a functor structure using the relation lifting:

$$\mathrm{Pred}(F)(P) = \coprod_{\pi_1}(\mathrm{Rel}(F)(\coprod_{\delta}(P))) = \coprod_{\pi_2}(\mathrm{Rel}(F)(\coprod_{\delta}(P))),$$

where $\delta = \langle id, id\rangle$ and $\coprod_f(X)$ is the image of X under f, so $\coprod_{\delta_x}(P) = \{(x, x) \mid x \in P\}$, $\coprod_{\pi_1}(R) = \{x_1 \mid \exists x_2.x_1 R x_2\}$ is the domain of the relation R, and $\coprod_{\pi_2}(R) = \{x_2 \mid \exists x_1.x_1 R x_2\}$ is its codomain.

The class of polynomial endofunctors is defined as the least class of endofunctors on **Sets** such that it contains the identity and constant functors, and is closed under product, coproduct, constant exponentiation, powerset and finite sequences. For polynomial endofunctors, $\mathrm{Rel}(F)$ and $\mathrm{Pred}(F)$ can be defined by induction on the structure of F. For further details on these definitions see [9]; we will introduce some of those when needed. For example, for the cases of labelled transition systems and Kripke structures we have:

$$\mathrm{Rel}(\mathcal{P}(id)^A)(R) = \{(f,g) \mid \forall a \in A.\, (f(a), g(a)) \in \{(U,V) \mid \forall u \in U.\, \exists v \in V.\, uRv \wedge \forall v \in V.\, \exists u \in U.\, uRv\}\}$$

$$\mathrm{Pred}(\mathcal{P}(id)^A)(P) = \{f \mid \forall a \in A.\, f(a) \in \{U \mid \forall u \in U.\, Pu\}\}$$

$$\mathrm{Rel}(\mathcal{P}(AP) \times \mathcal{P}(id))(R) = \{((u_1,u_2),(v_1,v_2)) \mid (u_1 = v_1.\, u_1, v_1 \in \mathcal{P}(AP)) \wedge (u_2,v_2) \in \{(U,V) \mid \forall u \in U.\, \exists v \in V.\, uRv \wedge \forall v \in V.\, \exists u \in U.\, uRv\}\}$$

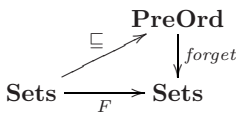
$$\mathrm{Pred}(\mathcal{P}(AP) \times \mathcal{P}(id))(P) = \{(u,v) \mid (u \subseteq \mathcal{P}(AP)) \wedge (v \in \{U \mid \forall u \in U.\, Pu\})\}$$

A bisimulation for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ which is "closed under c and d":

$$\text{if } (x,y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)(R)\,.$$

In the same way, an invariant for a coalgebra $c : X \longrightarrow FX$ is a predicate $P \subseteq X$ such that it is "closed under c", that is, if $x \in P$ then $c(x) \in \mathrm{Pred}(F)(P)$.

We will use the definition of simulation introduced by Hughes and Jacobs in [8] which uses an order $\sqsubseteq$ for functors F that makes the following diagram commute



Given an order $\sqsubseteq$ on F, a simulation for the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ such that

$$\text{if } (x,y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)_{\sqsubseteq}(R)\,,$$

where $\mathrm{Rel}(F)_{\sqsubseteq}(R)$ is defined as

$$\mathrm{Rel}(F)_{\sqsubseteq}(R) = \{(u,v) \mid \exists w \in F(R).\, u \sqsubseteq Fr_1(w) \wedge Fr_2(w) \sqsubseteq v\}\,.$$

To express properties we will use a generalization of LTL proposed by Jacobs (see [9]) that applies to arbitrary coalgebras, whose formulas are given by the following BNF expression:

$$\varphi = P \subseteq X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi\, \mathcal{U}\, \varphi$$

$\bigcirc$ is the *nexttime* operator and its semantics (abusing notation) is defined as $\bigcirc P = c^{-1}(\mathrm{Pred}(F)(P)) = \{x \in X \mid c(x) \in \mathrm{Pred}(F)(P)\}$; $\Box$ is the *henceforth* operator defined as $\Box P$ if exists an invariant for c, such that $Q \subseteq P$ with $x \in Q$ or, equivalently by means of the greatest fixed point ν, $\Box P = \nu S.(P \wedge \bigcirc S)$; $\Diamond$ is the *eventually* operator defined as $\Diamond P = \neg\Box\neg P$; and $\mathcal{U}$ is the *until* operator defined as $P\,\mathcal{U}\,Q = \mu S.(Q \vee (P \wedge \neg \bigcirc \neg S))$, where μ is the least fixed point.

We denote the set of states in X that satisfies φ as $[\![\varphi]\!]_X$. That is, if $P \subseteq X$ is a predicate, then $[\![P]\!]_X = P$; if $\alpha \in \{\neg, \bigcirc, \Box, \Diamond\}$ then $[\![\alpha\varphi]\!]_X = \alpha[\![\varphi]\!]_X$, and if $\beta \in \{\wedge, \vee, \Rightarrow, \mathcal{U}\}$ then $[\![\varphi_1\beta\varphi_2]\!]_X = [\![\varphi_1]\!]_X \beta [\![\varphi_2]\!]_X$. We will usually omit the reference to the set X when it is clear from the context. We say that an element x satisfies a formula φ, and we denote it by $c, x \models \varphi$, when $x \in [\![\varphi]\!]$. Again, we will usually omit the reference to the coalgebra c.

3 Reflection and Preservation in Bisimulations

These definitions of reflection and preservation are slightly more involved than for classical LTL because the logic proposed by Jacobs does not use atomic propositions, but predicates (subsets of the set of states). Later, we will see how atomic propositions can be introduced in the logic.

Given a predicate P on X and a binary relation $R \subseteq X \times Y$, we will say that an element $y \in Y$ is in the direct image of P, and we will denote it by $y \in RP$, if there exists $x \in X$ with $x \in P$ and xRy. The inverse image of R is just the direct image for the relation R^{-1}.

Definition 1. *Given two formulas φ on X and ψ on Y, built over predicates $P_1, \ldots P_n$ and $Q_1, \ldots Q_n$, respectively, and a binary relation $R \subseteq X \times Y$, we define the image of φ as a formula φ^* on Y, obtained by substituting in φ RP_i for P_i. Likewise, we build ψ^{-1}, the inverse of ψ, substituting $R^{-1}Q_i$ for Q_i in ψ.*

Remark 1. It is important to notice that φ^* coincides with φ^{-1} when we consider R^{-1} instead of R. Analogously, φ^{-1} is just φ^* when we consider R^{-1} instead of R.

Now we can define when a relation preserves or reflects properties.

Definition 2. *Let $R \subseteq X \times Y$ be a binary relation and a and b elements such that aRb. We say that R* preserves *the property φ on X if, whenever $a \models \varphi$, $b \models \varphi^*$. We say that R* reflects *the property φ on Y if $b \models \varphi$ implies $a \models \varphi^{-1}$.*

Let us first state a couple of technical lemmas whose proofs appear in [6].

Lemma 1. *Let F be a polynomial functor, $R \subseteq X \times Y$ a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, $P \subseteq Y$, $Q \subseteq X$ and xRy. If $d(y) \in \mathrm{Pred}(F)(P)$, then $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$; and if $c(x) \in \mathrm{Pred}(F)(Q)$, then $d(y) \in \mathrm{Pred}(F)(RQ)$.*

Another auxiliary lemma we need to prove the main result of this section is the following:

Lemma 2. *The direct and inverse images of an invariant are also invariants.*

Proof. Let R be a bisimulation between $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Let us suppose that $P \subseteq X$ is an invariant and let us prove that so is RP; that is, for all $y \in RP$ it must be the case that $d(y) \in \mathrm{Pred}(F)(RP)$. If $y \in RP$, then there exists $x \in P$ such that xRy. Since P is an invariant, we also have $c(x) \in \mathrm{Pred}(F)(P)$ and by Lemma 1 we get $d(y) \in \mathrm{Pred}(F)(RP)$.

On the other hand, since R^{-1} is also a bisimulation, the inverse image of an invariant is an invariant too. □

At this point it is interesting to recall that our objective is to prove that bisimulations preserve and reflect properties of a temporal logic, that is, if we have xRy and $y \models \varphi$ then we must also have $x \models \varphi^{-1}$; and, analogously, if $x \models \varphi$ then $y \models \varphi^*$. We will show this result for all temporal operators except for the negation; it is well-known that negation is reflected and preserved by standard bisimulations, but not here because of the lack of atomic propositions in the coalgebraic temporal logic.

To prove the result for the rest of temporal operators, we will see that if $y \in [\![\varphi]\!]$ then we also have $x \in R^{-1}[\![\varphi]\!]$ and, analogously, if $x \in [\![\varphi]\!]$ then $y \in R[\![\varphi]\!]$. Ideally, we would like to have both $R^{-1}[\![\varphi]\!] = [\![\varphi^{-1}]\!]$ and $R[\![\varphi]\!] = [\![\varphi^*]\!]$ but, in general, only the inclusion $\subseteq$ is true. Fortunately this is enough to prove our propositions, since the temporal operators are all monotonic except for the negation. In fact, here is where the problem with negation appears.

Lemma 3 ([6]). *Let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. For all temporal formulas φ and ψ which do not contain the negation operator, it follows that*

$$R^{-1}[\![\varphi]\!]_Y \subseteq [\![\varphi^{-1}]\!]_X \quad \textit{and} \quad R[\![\psi]\!]_X \subseteq [\![\psi^*]\!]_Y \,.$$

Finally we can show that bisimulations reflect and preserve properties given by any temporal operator except for the negation.

Proposition 1. *Let ψ be a formula over a set Y which does not use the* negation *operator and let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then the property ψ is reflected by R.*

Proof. The result is proved by structural induction over the formula ψ using the first half of Lemmas 1 and 3, and Lemma 2. See [6] for further details. □

Preservation of properties is a consequence of the reflection of properties together with the fact that if R is a bisimulation then R^{-1} is also a bisimulation. We have thus proved the following theorem.

Theorem 1. *Let ψ and φ be formulas over sets Y and X, respectively, which do not use the* negation *operator and let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then ψ is reflected by R and φ is preserved by R.*

4 Reflection and Preservation in Simulations

In [3,16] it is proved not only that bisimulations reflect and preserve properties but also that simulations reflect them: it turns out that this result does not generalize straightforwardly to the coalgebraic setting.

The main problem that we have found concerning this is that the coalgebraic definition of simulation uses an arbitrary functorial order $\sqsubseteq$, and in general reflection of properties will not hold for all orders.

Let us show a counterexample that will convince us that simulations may not reflect properties without restricting the orders. Let us take $F = \mathcal{P}(id)$, $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$ and the coalgebras c and d defined as $c(x_1) = \{x_1, x_2\}$, $c(x_2) = \{x_2\}$, $d(y_1) = y_2$ and $d(y_2) = y_2$. We define $u \sqsubseteq v$ whenever $v \subseteq u$ and consider the formula $\varphi = \bigcirc P$, where $P = \{y_2\}$, and the simulation $R = \{(x_1, y_2)\}$. It is immediate to check that R is a simulation and $y_2 \in [\![\varphi]\!]$, but $x_1 \notin [\![\varphi^{-1}]\!]$.

- $y_2 \in [\![\varphi]\!]$. Indeed, since $d(y_2) = y_2$ then $y_2 \in [\![\varphi]\!] = \bigcirc P$ is equivalent to $y_2 \in P = \{y_2\}$, which is trivially true.
- $x_1 \notin [\![\varphi^{-1}]\!]$. By definition, $\varphi^{-1} = \bigcirc R^{-1} P = \bigcirc\{x_1\}$. Since $c(x_1) = \{x_1, x_2\}$, it is enough to see that $x_2 \notin \{x_1\}$, which is also true.

As a consequence, we will need to restrict the functorial orders that are involved in the definition of simulation. In a first approach we will impose an extra requirement that the order must fulfill, and later we will not only restrict the orders but also the functors that are involved.

4.1 Restricting the Orders

The idea is that we are going to require an extra property for each pair of elements which are related by the order. In particular, we are particularly interested in the following property (which is defined in [8]):

Definition 3. *Given a functor F : **Sets** $\longrightarrow$ **Sets**, we say that an order $\sqsubseteq$ associated to it is "down-closed" whenever $a \sqsubseteq b$, with $a, b \in FX$, implies that*

$$b \in \mathrm{Pred}(F)(P) \implies a \in \mathrm{Pred}(F)(P), \quad \textit{for all predicates } P \subseteq X\,.$$

We can show some examples of down-closed orders:

Example 2. 1. Kripke structures are defined by the functor $F = \mathcal{P}(AP) \times \mathcal{P}(id)$, so a down-closed order must fulfill that if $(u, v) \sqsubseteq (u', v')$, then $(u', v') \in \mathrm{Pred}(F)(P)$ implies $(u, v) \in \mathrm{Pred}(F)(P)$; that is, by definition of $\mathrm{Pred}(\mathcal{P}(AP) \times \mathcal{P}(id))$, $u, u' \subseteq \mathcal{P}(AP)$ and, if $v' \in \mathrm{Pred}(\mathcal{P}(id))(P) = \{U \mid \forall u \in U.\, u \in P\}$ then $v \in \mathrm{Pred}(\mathcal{P}(id))(P)$. In other words, for all $b \in v$ and $b' \in v'$, if $b' \in P$ then $b \in P$. Therefore, what is needed in this case is that the set of successors v of the smaller pair is contained in the set of successors v' of the bigger pair, that is, if $(u, v) \sqsubseteq (u', v')$ then $v \subseteq v'$.

2. Labelled transition systems are defined by the functor $F = \mathcal{P}(id)^A$, so the order must fulfill the following: if $u \sqsubseteq v$ then $\forall a \in A.\ u(a) \subseteq u'(a)$.

Those examples show that there are not many down-closed orders, but it does not seem clear how to further extend this class in such a way that we could still prove the reflection of properties by simulations. Unfortunately, even under this restriction we can only prove reflection (or preservation) of formulas that only use the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$.

To convince us of this fact, we present a counterexample with operator $\Diamond$. Let $X = \{x_1, x_2\}, Y = \{y_1, y_2\}$ and the functor $F = \mathcal{P}(id)$. We consider the following down-closed order: $u \sqsubseteq v$ if $u \subseteq v$. We also define the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ as $c(x_1) = \{x_1\}$, $c(x_2) = \{x_2\}$, $d(y_1) = \{y_1, y_2\}$ and $d(y_2) = \{y_2\}$. Obviously $R = \{(x_1, y_1)\}$ is a simulation since $c(x_1) = \{x_1\} \sqsubseteq \{x_1\}$ and $\{y_1\} \sqsubseteq \{y_1, y_2\} = d(y_1)$ and, also, $\{x_1\}\mathrm{Rel}(F)(R)\{y_1\}$. We have $y_1 \in \Diamond\{y_2\}$, since we can reach y_2 from y_1, but $x_1 \notin \Diamond R^{-1}\{y_2\} = \Diamond\emptyset$. Indeed, $x_1 \notin \Diamond\emptyset$ is equivalent to $x_1 \in \Box\neg\emptyset$ and this is true since $\{x_1\}$ is an invariant such that $x_1 \in \{x_1\}$, with $\{x_1\} \subseteq \neg\emptyset$.

In order to prove reflection of properties that only use the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, we will need a previous elementary result involving binary relations.

Proposition 2. *Let $R \subseteq X \times Y$ be a binary relation and $P \subseteq Y$ a predicate. Let us suppose that $u\mathrm{Rel}(F)(R)v$; then, if $v \in \mathrm{Pred}(F)(P)$ it is also true that $u \in \mathrm{Pred}(F)(R^{-1}P)$.*

Proof. Once again the proof will proceed by structural induction on the functor F. See [6] for further details. □

We will also need a subtle adaptation of Lemmas 2 and 3 from the framework of bisimulations to the framework of simulations. In particular, we can adapt Lemma 2 to prove that if Q is an invariant and R a simulation, $R^{-1}Q$ is still an invariant, whereas the first half of Lemma 3 will also be true in the framework of simulations for formulas that only use the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$.

Lemma 4. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, with a down-closed order, and let $Q \subseteq Y$ be an invariant. Then $R^{-1}Q$ is also an invariant.*

Proof. We are going to show that for all $x \in R^{-1}Q$ we have $c(x) \in \mathrm{Pred}(F)(R^{-1}Q)$. Let us take an arbitrary $x \in R^{-1}Q$; then, by definition there exists $y \in Q$ such that xRy and, since Q is an invariant, $d(y) \in \mathrm{Pred}(F)(Q)$. On the other hand, since R is a simulation, $c(x) \sqsubseteq u\mathrm{Rel}(F)(R)v \sqsubseteq d(y)$. Henceforth, since we are working with a down-closed order and $d(y) \in \mathrm{Pred}(F)(Q)$, then $v \in \mathrm{Pred}(F)(Q)$. Also, by Proposition 2 we have $u \in \mathrm{Pred}(F)(R^{-1}Q)$ and, using again that the order is down-closed, it follows that $c(x) \in \mathrm{Pred}(F)(R^{-1}Q)$. □

Lemma 5 ([6]). *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, with a down-closed order. If φ is a temporal formula constructed only with operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, then*

$$R^{-1}[\![\varphi]\!]_Y \subseteq [\![\varphi^{-1}]\!]_X .$$

Now we can state the corresponding theorem:

Theorem 2 ([6]). *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ with a down-closed order. If φ is a temporal formula constructed only with operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, then the property φ is reflected by the simulation.*

Instead of considering down-closed orders, we could have imposed the converse implication, that is, those orders that satisfy that if $a \in \mathrm{Pred}(F)(P)$ then $b \in \mathrm{Pred}(F)(P)$.

Definition 4. *Given a functor F : **Sets** $\longrightarrow$ **Sets** we say that an order $\sqsubseteq$ is* up-closed *if whenever $a \sqsubseteq b$ then*

$$a \in \mathrm{Pred}(F)(P) \implies b \in \mathrm{Pred}(F)(P), \quad \textit{for all predicates } P\,.$$

Obviously up-closed is symmetrical to down-closed, that is, it is equivalent to taking $\sqsubseteq^{op}$ instead of $\sqsubseteq$ in Definition 3. So, for example, in the case of Kripke structures an up-closed order would satisfy $(u, v) \sqsubseteq (u', v')$ if $v' \subseteq v$.

The interesting thing about up-closed orders is that they allow us to prove *preservation* of properties; again, this result will hold only for formulas constructed with the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$. We need the following auxiliary result whose proof is analogous to the case of down-closed orders. Since if R is a simulation for the order $\sqsubseteq$, then R^{-1} is a simulation for the oposite order $\sqsubseteq^{op}$, we can apply Theorem 2 to get the following (see [6] for more details):

Theorem 3. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ carrying an up-closed order. If φ is a temporal formula constructed only with the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, then R preserves the property φ.*

4.2 Restricting the Class of Functors

As we have just seen, it is not enough to restrict ourselves to down-closed (or up-closed) orders to get a valid result for all properties. What we want is a necessary and sufficient condition over functorial orders that implies reflection (or preservation) of properties by simulations. So far we have not found such a condition, but we have a sufficient one for simulations to reflect properties (and, in fact, also so that they preserve properties).

Recalling the structure of lemmas and propositions used to prove reflection and preservation of properties by bisimulations, we notice that the key ingredient was Lemma 1. With this lemma we were able to prove directly preservation of invariants (Lemma 2) and the relation between R^{-1} (respectively R) of a formula and the inverse of a formula (respectively direct image of a formula). Also, Lemma 1 was essential to prove directly reflection and preservation of formulas built with the *nexttime* operator and the rest of temporal operators.

In the previous section the problem we faced was that either the second half of Lemma 1 (for down-closed orders) or the first half of Lemma 1 (for up-closed

orders) held, but not both simultaneously. As a consequence, the results for the operators *eventually* and *until* did not hold. So, if we were capable of finding a subclass of functors and orders such that they fulfill results analogous to Lemma 1 then, translating those proofs, we would get reflection and preservation of arbitrary properties.

We are going to define a subclass of functors and orders in the way that Hughes and Jacobs did in [8] for the subclass **Poly**.

Definition 5. *The class* **Order** *is the least class of functors closed under the following operations:*

1. *For every preorder* $(A, \leq)$*, the constant functor* $X \mapsto A$ *with the order given by* $\sqsubseteq_X = \leq_A$.
2. *The identity functor with equality order.*
3. *Given two polynomial functors* F_1 *and* F_2 *with orders* $\sqsubseteq^1$ *and* $\sqsubseteq^2$*, the product functor* $F_1 \times F_2$ *with order* $\sqsubseteq_X$ *given by*

$$(u, v) \sqsubseteq_X (u', v') \quad \text{if} \quad u \sqsubseteq^1 u' \quad \text{and} \quad v \sqsubseteq^2 v' .$$

4. *Given the polynomial functor* F *with order* $\sqsubseteq^F$ *and the set* A*, the functor* F^A *with order* $\sqsubseteq_X$ *given by*

$$u \sqsubseteq_X v \text{ if } u(a) \sqsubseteq^F v(a) \text{ for all } a \in A.$$

5. *Given two polynomial functors* F_1 *and* F_2 *with orders* $\sqsubseteq^1$ *and* $\sqsubseteq^2$*, the coproduct functor* $F_1 + F_2$ *with order* $\sqsubseteq_X$ *given by*

$$\begin{aligned} u \sqsubseteq_X v \text{ if } u = \kappa_1(u_0) \text{ and } v = \kappa_1(v_0) \text{ with } u_0 \sqsubseteq^1 v_0 \\ \text{or } u = \kappa_2(u_0) \text{ and } v = \kappa_2(v_0) \text{ with } u_0 \sqsubseteq^2 v_0 . \end{aligned}$$

6. *Given the polynomial functor* F *with order* $\sqsubseteq^F$*, the powerset functor* $\mathcal{P}(F)$ *with order* $\sqsubseteq_X$ *given by*

$$\begin{aligned} u \sqsubseteq_X v \quad \text{if} \quad \forall a \in u\ \exists b \in v \quad \text{such that} \quad a \sqsubseteq^F b \\ \text{and also } \forall b \in v\ \exists a \in u \quad \text{such that} \quad a \sqsubseteq^F b . \end{aligned}$$

For example the usual order for Kripke structures is not in the class **Order**. Besides, in the definition of **Poly** in [8] the authors did not consider the powerset functor but we do, although we are not using the *usual* order for this functor.

At first, to obtain that simulations not only reflect but also preserve properties may seem a little surprising. If we think about the elements in the subclass **Order** we notice that we have restricted the orders to equality-like orders, that is, almost all possible orders in **Order** are the equality. However, since the class **Order** is very similar to the class **Poly**, it has the same good properties shown in [8] (like the stablility of the orders and functors).

Example 3. 1. If we consider the functor $\mathcal{P}(id)$, then the order $\sqsubseteq$ defined in Definition 5 says that $u \sqsubseteq v$ if and only if for each $a \in u$ there exists $b \in v$ such that $a = b$, and if for each $b \in v$ there exists $a \in u$ such that $a = b$. This means that $\sqsubseteq$ is the identity relation. As an immediate consequence for transition systems the only possible **Order** simulations are bisimulations.

2. If we consider the functor $A \times id$ where A has a preorder $\leq_A$ different from the identity, the order $\sqsubseteq$ from Definition 5 is the following: $(u, v) \sqsubseteq (u', v')$ iff $v = v'$ and $u \leq_A u'$. So, if $\leq_A$ is not the identity, neither is $\sqsubseteq$. For example, let us take $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2\}$, $AP = \{p_1, p_2, p_3\}$ and consider the functor $F = \mathcal{P}(id) \times \mathcal{P}(AP)$ and the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ defined by $c(x_1) = (\{x_2, x_3\}, \{p_1\})$, $c(x_2) = (\{x_3\}, \{p_2\})$, $c(x_3) = (\{x_2\}, \{p_3\})$, $d(y_1) = (\{y_2\}, \{p_2\})$ and $d(y_2) = (\{y_2\}, \{p_1\})$. Obviously there is no bisimulation between x_1 and y_1 since this atomic propositions are not the same, but taking the order $\sqsubseteq$ defined as $(u, v) \sqsubseteq (u', v')$ iff $u = u'$ (that is, taking as the preorder $\leq_{AP}$ the total relation) we have that there exists a simulation R in **Order** between x_1 and y_1.

Lemma 6 ([6]). *Let $R \subseteq X \times Y$ be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, such that the functor F is in the class* **Order**. *Let us also suppose that $P \subseteq Y$ and xRy; then, if $d(y) \in \mathrm{Pred}(F)(P)$ we have $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.*

In a similar way we have the corresponding lemma involving direct predicates.

Lemma 7. *Let $R \subseteq X \times Y$ be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, such that the functor F is in* **Order**. *Let us suppose also that $P \subseteq X$ and xRy. Then, if $c(x) \in \mathrm{Pred}(F)(P)$, $d(y) \in \mathrm{Pred}(F)(RP)$.*

Now we can conclude that under these hypothesis simulations reflect and preserve properties, simultaneously! This fact is a straightforward result from Lemmas 6 and 7.

Theorem 4. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, with F a polynomial functor in the class* **Order**. *Then, the simulation R reflects and preserves properties.*

5 Including Atomic Propositions

A consequence of the fact that the logic proposed by Jacobs does not introduce atomic propositions was the need of giving non-standard definitions of reflection and preservation of properties. Kurz, in his work [13] includes atomic propositions in a temporal logic for coalgebras by means of natural transformations.

Definition 6. *Given a set AP of atomic propositions, the formulas of the temporal logic associated to a coalgebra $c : X \longrightarrow FX$ are given by the BNF expression:*

$$\varphi = p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi \,\mathcal{U}\, \varphi$$

where $p \in AP$ is an atomic proposition.

Kurz also defines when a state x satisfies an atomic proposition p, that is, he defines the semantics of an atomic proposition.

Definition 7. *Let F : **Sets** $\longrightarrow$ **Sets** be a functor and AP a set of atomic propositions. Let $\nu : F \Rightarrow \mathcal{P}(AP)$ be a natural transformation and $c : X \longrightarrow FX$ a coalgebra. We say that x satisfies an atomic proposition $p \in AP$, and denote it $x \models p$, when $p \in (\nu_X \circ c)(x)$. This way $[\![p]\!] = \{x \mid p \in (\nu_X \circ c)(x)\}$.*

Notice that in fact this defines not only a semantics but a family of possible semantics that depends on the natural transformation. For example, we can define a natural transformation for the functor for Kripke structures in this way:

$$\begin{array}{ccc} \nu_X : \mathcal{P}(AP) \times \mathcal{P}(X) & \longrightarrow & \mathcal{P}(AP) \\ (P, Q) & \longmapsto & P \end{array}$$

With ν_X we have characterized the standard semantics of LTL for Kripke structures. Analogously, we could define the following interpretation: $\nu'_X(P, Q) = \mathcal{P}(AP) \setminus P$.

Introducing in our temporal logic the semantics of the atomic propositions, we can prove the following theorem involving bisimulations:

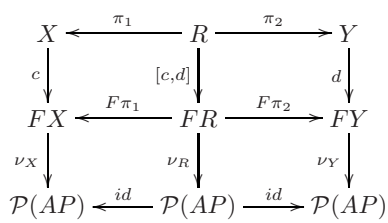
Theorem 5. *Let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Let φ be a temporal formula; then, the following is true for all $x \in X$ and $y \in Y$ such that xRy:*

$$x \in [\![\varphi]\!]_X \quad \Longleftrightarrow \quad y \in [\![\varphi]\!]_Y \, .$$

Here we have captured in the same theorem the classical ideas of reflection and preservation of properties: we have some property in the lefthand side of a bisimulation if and only if we have the property in its righthand side. In this case the theorem is true also for the negation operator thanks to the atomic propositions. Intuitively, this is because now we have an "if and only if" theorem, whereas in Theorem 1 we needed to reason separately for each implication using monotonicity, and negation lacks it. Also notice that even though we could think that in Theorem 1 our predicates played the role of atomic propositions, there are some essential differences: first, predicates are not independent of each other, unlike atomic propositions, and secondly, while atomic propositions stay the same predicates vary with each set of states.

Proof. Once again the proof will proceed by structural induction on the formula φ. We only show some of the cases (the complete proof can be found in [6]).

1. Let $\varphi = p$ where p is an arbitrary atomic proposition. This way we have the following diagram, for ν an arbitrary natural trasformation:



This diagram is commutative. Indeed, since R is a bisimulation the upper side commutes, while the lower side commutes because ν is a natural transformation.
So, $x \in [\![\varphi]\!]_X$ means by definition that $p \in (\nu_X \circ c)(x)$. Since the diagram commutes then $p \in (\nu_R \circ [c,d])(x,y) \Leftrightarrow p \in (\nu_Y \circ d)(y)$, that is, $y \in [\![\varphi]\!]_Y$.

2. Let us suppose $\varphi = \neg\varphi_0$. In this case we must show that $x \in \neg[\![\varphi_0]\!]_X$ if and only if $y \in \neg[\![\varphi_0]\!]_Y$, that is, we must see that $x \notin [\![\varphi_0]\!]_X$ if and only if $y \notin [\![\varphi_0]\!]_Y$. By induction hypothesis we have $x \in [\![\varphi_0]\!]_X$ if and only if $y \in [\![\varphi_0]\!]_Y$.
3. Let us suppose now that $\varphi = \bigcirc\varphi_0$. We must prove that $x \in \bigcirc[\![\varphi_0]\!]_X$ is equivalent to $y \in \bigcirc[\![\varphi_0]\!]_Y$, that is, $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$ is equivalent to $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$. The latter will be proved by structural induction on the functor F. As an example we show the case of $F = G^A$. Let us prove only one implication since the other one is almost identical. We have

$$\mathrm{Pred}(F)([\![\varphi_0]\!]_X) = \{f \mid \forall a \in A.\ f(a) \in \mathrm{Pred}(G)([\![\varphi_0]\!]_X)\}\,.$$

Once again, as we have shown in other proofs, we define for each $a \in A$ and each F-coalgebra $c : X \longrightarrow F(X)$ a G-coalgebra, $c^a : X \longrightarrow G(X)$ where for each $x \in X$ we have $c^a(x) = c(x)(a)$. In this way, we have xRy and $c^a(x) = c(x)(a) \in \mathrm{Pred}(G)([\![\varphi_0]\!]_X)$. By induction hypothesis we have that $d^a(y) \in \mathrm{Pred}(G)([\![\varphi_0]\!]_Y)$. Since this is a valid argument for all $a \in A$, we obtain $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$.
4. $\varphi = \Box\varphi_0$. Assuming that $x \in [\![\varphi]\!]_X$ we get that there exists

$$Q \subseteq X \text{ an invariant for } c \text{ with } Q \subseteq [\![\varphi_0]\!]_X \text{ and } x \in Q.$$

Now, RQ is a invariant for d and, also, such that $RQ \subseteq [\![\varphi_0]\!]_Y$ with $y \in RQ$. Indeed, if $x \in Q$ then $y \in RQ$ and if $b \in RQ$ there must exists some $a \in Q \subseteq [\![\varphi_0]\!]_X$ such that aRb. So, by induction hypothesis we get that $b \in [\![\varphi_0]\!]_Y$
On the other hand, if $y \in [\![\varphi]\!]_Y$ there must exists some invariant T on Y, such that $T \subseteq [\![\varphi_0]\!]_Y$ with $y \in T$, hence for proving $x \in [\![\varphi]\!]_X$ it is enough to consider the invariant $R^{-1}T$. □

To obtain a similar result for simulations, we will need again to restrict the class of functors and orders as we did in Sections 4.1 and 4.2. In particular we are interested in the following antimonotonicity property: if $u \sqsubseteq u'$ then $\nu(u') \subseteq \nu(u)$.

Definition 8. *Let F : **Sets** $\longrightarrow$ **Sets** be a functor, AP a set of atomic propositions and $\nu : F \Rightarrow \mathcal{P}(AP)$ a natural transformation. We say that $\sqsubseteq$ is a down-natural ν-order if, whenever $u \sqsubseteq u'$ then $\nu(u') \subseteq \nu(u)$.*

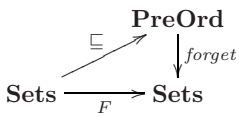
Obviously this definition depends on the natural transformation that we consider in each case. For example, for Kripke structures we have the following natural transformation: $\nu_X((A_X, B_X)) = A_X \subseteq AP$. To obtain a down-natural ν-order

the following must hold: $(u, v) \sqsubseteq (u', v')$ then $\nu((u', v')) \subseteq \nu((u, v))$, that is, it will be enough to require $(u, v) \sqsubseteq (u', v')$ iff $u' \subseteq u$.

This way, if we combine the down-closed and the down-natural orders we get:

$$\text{If} \quad (u, v) \sqsubseteq (u', v') \quad \text{then} \quad u' \subseteq u \text{ and } v \subseteq v' \,.$$

This characterization is not as restrictive as one could think. Indeed, if we recall the definition of functorial order we had:



This diagram means that the functor F and the order $\sqsubseteq$ almost have the same structure and indeed, we could use a natural transformation between $\sqsubseteq$ and $\mathcal{P}(AP)$ in Definition 7 instead of a natural transformation between F and $\mathcal{P}(AP)$, that is, $\nu : \sqsubseteq \Rightarrow \mathcal{P}(AP)$. Considering ν in this way, an immediate consequence is that if we take as order in $\mathcal{P}(AP)$ the relation $\supseteq$ (as is done in [16]), then $u \sqsubseteq v$ implies $\nu(u) \sqsubseteq \nu(v)$.

We can tackle the proof of reflection of properties (with atomic propositions) by simulations as we did in Section 4.1, imposing to the order not only to be down-natural but also down-closed. But, if we do that we will find the same difficulties we faced in Section 4.1 (that is, we would not be able to prove reflection of formulas built with the operators *until* and *eventually*). Therefore, we must restrict the class of functors and orders, as we did with the class **Order** in Section 4.2, but imposing also that the orders must be down-natural.

Definition 9. *The class* **Down-Natural ν-Order** *is the subclass of* **Order** *where all orders are down-natural.*

Notice that we are defining a different class for each natural transformation ν. Under this condition we state the corresponding theorem involving simulations and the reflection of properties (with atomic propositions); for the proof see [6].

Theorem 6. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ on the same polynomial functor F from* **Sets** *to* **Sets** *belonging to the class* **Down-Natural ν-Order** *and let φ be a temporal formula. Then, for each $x \in X$ and $y \in Y$ such that xRy:*

$$y \in [\![\varphi]\!]_Y \quad \Longrightarrow \quad x \in [\![\varphi]\!]_X \,.$$

We showed above that simulations for functors in the class **Order** reflected and preserved all kinds of properties. Instead, now we can only prove one implication, that corresponding to the reflection of properties. This is so because down-natural ν-orders have a natural direction.

Exactly in the same way as we did with down-natural ν-orders, we can define the corresponding class of up-natural ν-orders:

Definition 10. *Let $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ be a functor, AP a set of atomic propositions and $\nu : F \Rightarrow \mathcal{P}(AP)$ a natural transformation. We say that $\sqsubseteq$ is an up-natural ν-order if $u \sqsubseteq u'$ implies $\nu(u) \subseteq \nu(u')$.*

As we did for down-natural ν-orders, we define a subclass of **Order**:

Definition 11. *The class* **Up-Natural ν-Order** *is the subclass of* **Order** *where all orders are up-natural.*

Theorem 7. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ on the same polynomial functor F in the class* **Up-Natural ν-Order**, *and let φ be a temporal formula. Then, for all $x \in X$ and $y \in Y$ such that xRy:*

$$x \in [\![\varphi]\!]_X \quad \Longrightarrow \quad y \in [\![\varphi]\!]_Y \, .$$

6 Conclusions

The main goal of this paper was to study under what assumptions coalgebraic simulations reflect properties. In our way towards the proof of this result, we were also able to prove reflection and preservation of properties by coalgebraic bisimulations. For expressing the properties we used Jacobs' temporal logic [9], later extended with atomic propositions using the idea presented in [13].

That coalgebraic bisimulations reflect and preserve properties expressed in modal logic is a well-known topic (e.g, [10,13,17]), but not so the corresponding results for simulations. The main difficulty is that Hughes and Jacobs' notion of simulation is defined by means of an arbitrary functorial order which bestows them with a high degree of freedom. We have dealt with this by restricting the class of functorial orders (although even so we are not able of obtaining a general result) and by restricting also the class of allowed functors.

In order to get more general results on the subject, an interesting path that we intend to explore is the search for a canonical notion of simulation. This definition would provide us, not only with a "natural" way to understand simulations but, hopefully, would also give rise to "natural" general results about reflection of properties.

Another promising direction of research is the study of reflection and preservation of properties in probabilistic systems, following our results of [4] in combination with the ideas presented in [7,5,2].

Acknowledgement

The authors would like to thank the anonymous referees for their comments and suggestions.

References

1. Aczel, P., Mendler, N.P.: A final coalgebra theorem. In: Dybjer, P., Pitts, A.M., Pitt, D.H., Poigné, A., Rydeheard, D.E. (eds.) Category Theory and Computer Science. LNCS, vol. 389, pp. 357–365. Springer, Heidelberg (1989)
2. Bartels, F., Sokolova, A., de Vink, E.P.: A hierarchy of probabilistic system types. Theor. Comput. Sci. 327(1-2), 3–22 (2004)
3. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
4. de Frutos Escrig, D., Palomino, M., Fábregas, I.: Multiset bisimulation as a common framework for ordinary and probabilistic bisimulations (submitted)
5. de Vink, E.P., Rutten, J.J.M.M.: Bisimulation for probabilistic transition systems: a coalgebraic approach. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 4460–4470. Springer, Heidelberg (1997)
6. Fábregas, I., Palomino, M., de Frutos Escrig, D.: Reflection and preservation of properties in coalgebraic (bi)simulations (extended) (2007), `http://maude.sip.ucm.es/~miguelpt/`
7. Hasuo, I.: Generic forward and backward simulations. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 406–420. Springer, Heidelberg (2006)
8. Hughes, J., Jacobs, B.: Simulations in coalgebra. Theor. Comput. Sci. 327(1-2), 71–108 (2004)
9. Jacobs, B.: Introduction to Coalgebra. Towards Mathematics of States and Observations. Book in preparation. Draft available in the web, `http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`
10. Jacobs, B.: *Categorical Logic and Type Theory*. Studies in Logic and the Foundations of Mathematics, vol. 141. North-Holland, Amsterdam (1999)
11. Jacobs, B., Rutten, J.J.M.M.: A tutorial on (co)algebras and (co)induction. Bulletin of the European Association for Theoretical Computer Science 62, 222–259 (1997)
12. Kesten, Y., Pnueli, A.: Control and data abstraction: The cornerstones of practical formal verification. International Journal on Software Tools for Technology Transfer 4(2), 328–342 (2000)
13. Kurz, A.: Logics for coalgebras and applications to computer science. PhD thesis, Universität München (2000)
14. Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., Bensalem, S.: Property preserving abstractions for the verification of concurrent systems. Formal Methods in System Design 6, 1–36 (1995)
15. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
16. Palomino, M.: Reflexión, abstracción y simulación en la lógica de reescritura. PhD thesis, Universidad Complutense de Madrid, Spain (March 2005)
17. Pattinson, D.: Expressivity Results in the Modal Logic of Coalgebras. PhD thesis, Universität München (2001)
18. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theor. Comput. Sci. 249(1), 3–80 (2000)

Non-strongly Stable Orders Also Define Interesting Simulation Relations*

Ignacio Fábregas, David de Frutos Escrig, and Miguel Palomino

Departamento de Sistemas Informáticos y Computación, UCM
fabregas@fdi.ucm.es, {miguelpt,defrutos}@sip.ucm.es

Abstract. We present a study of the notion of coalgebraic simulation introduced by Hughes and Jacobs. Although in their original paper they allow any functorial order in their definition of coalgebraic simulation, for the simulation relations to have good properties they focus their attention on functors with orders which are strongly stable. This guarantees a so-called "composition-preserving" property from which all the desired good properties follow. We have noticed that the notion of strong stability not only ensures such good properties but also "distinguishes the direction" of the simulation. For example, the classic notion of simulation for labeled transition systems, the relation "p is simulated by q", can be defined as a coalgebraic simulation relation by means of a strongly stable order, whereas the opposite relation, "p simulates q", cannot. Our study was motivated by some interesting classes of simulations that illustrate the application of these results: covariant-contravariant simulations and conformance simulations.

1 Introduction and Presentation of Our New Results

Simulations are a very natural way to compare systems defined by transition systems or other related mechanisms based on the description of systems by means of the actions they can execute at each of their states [11]. They can be enriched in several ways to obtain, in particular, the important ready simulation semantics [2,8], as well as other more elaborated ones such as nested simulations [5]. Quite recently we have studied the general concept of constrained simulation [3], proving that all the simulation relations constrained by an adequate condition have similar properties. The semantics of these constrained simulations is also the basis for our unified presentation of the semantics of processes [4], where all the semantics in the ltbt-spectrum [13] (and other new semantics) are classified in a systematic way.

Hughes and Jacobs [6] have also developed a systematic study of simulation-like relations, this time in a purely coalgebraic context, so that simulations are studied in connection with bisimulations [11], the fundamental concept to define equivalence in the coalgebraic world. Their coalgebraic simulations are defined in terms of an order $\sqsubseteq$ associated to the functor F corresponding to the coalgebra $c : X \longrightarrow FX$ that we want to observe. In this way they obtain a very general notion of coalgebraic simulation, not only because all functors F are considered, including in particular the important

* Research supported by the Spanish projects DESAFIOS TIN2006-15660-C02-01, WEST TIN2006-15578-C02-01, PROMESAS S-0505/TIC/0407 and UCM-BSCH GR58/08/910606.

A. Kurz, M. Lenisa, and A. Tarlecki (Eds.): CALCO 2009, LNCS 5728, pp. 221–235, 2009.

class of polynomial functors, but also because by changing the family of orders $\sqsubseteq_X$ many different families of simulation relations can be obtained. The general properties of these simulations can be studied in the defined coalgebraic framework, thus avoiding the need of similar proofs for each of the particular classes of simulations.

Certainly, this generic presentation of the notion of coalgebraic simulation has as advantage that it provides a wide and abstract framework where one can try to isolate and take advantage of the main properties of all the simulation-like relations. However, at the same time it can be argued that the proposal fails to capture in a tight manner the spirit of simulation relations because, in addition to the natural notions of simulations, the framework also allows for other less interesting relations. This has as a result that some natural properties of simulations cannot be proved in general, simply due to the fact that they are not satisfied by all of the permitted coalgebraic simulation relations. For instance, the induced similarity relation between systems is not always an order because transitivity is not always satisfied. In order to guarantee transivity, and other related properties of coalgebraic simulations, Jacobs and Hughes introduce in [7] the composition-preserving property to the order $\sqsubseteq$ that induces the simulation relation. In [6] they continue with the study of the topic and present *stability* of orders as a natural categorical property to guarantee that an order is composition-preserving. They also comment that stability is not easy to check and introduce a stronger condition (that we will call right-stability) so that, whenever applicable, the checking of the main properties of coalgebraic simulations becomes much simpler than in the general case.

Roughly speaking, given an order $\sqsubseteq_X$ on FX for each set X, the induced coalgebraic simulations are defined in the same way as bisimulations for F, but allowing a double application of $\sqsubseteq$ on the two sides of the defined relation. More precisely, instead of the functor $\mathrm{Rel}(F)$ defining plain bisimulations, $\mathrm{Rel}_{\sqsubseteq}(F)$ defined as $\sqsubseteq_Y \circ \mathrm{Rel}(F) \circ \sqsubseteq_X$ is used. There are several interesting facts hidden behind the apparent simplicity of this definition. The first one is that, in general, it only defines an order and not an equivalence relation, even if it is based on bisimulations (that always define an equivalence relation, namely, bisimilarity). The reason is that the order $\sqsubseteq$ appears "in the same direction" on both sides of the definition, thus breaking its symmetry. However, we can also define some equivalence relations weaker than bisimilarity by using an equivalence relation $\equiv$ as the order $\sqsubseteq$. Another interesting fact is that whenever we define a coalgebraic simulation by using $\sqsubseteq$, the inverse order $\sqsupseteq$ defines the inverse relation of that defined by $\sqsubseteq$ once we also interchange the roles of the related sets X and Y (so we could say that we are defining in fact the same relation but looking at it from the other side). Stability is also a symmetric condition, so that whenever an order $\sqsubseteq$ on a functor F is stable, the inverse order $\sqsupseteq$ is stable for F, too. This is quite reasonable, since stability is imposed in order to guaratee transitivity of the generated similarity relation and the inverse of a transitive relation is also transitive, so that whenever $\sqsubseteq$ generates an "admissible" similarity relation (meaning that it is an order), the inverse order $\sqsupseteq$ must be also admissible.

It is worth noting that the stronger condition guaranteeing stability is asymmetric. In fact, Hughes and Jacobs prove in [6] that "right-stability" implies that

$$\mathrm{Rel}(F)(R) \circ \sqsubseteq_X \subseteq \sqsubseteq_Y \circ \mathrm{Rel}(F)(R), \tag{1}$$

which in fact motivates our name for the condition.

A second surprise was to notice that, in most cases, right-stability induces a "natural direction" on the orders defining the coalgebraic simulation. For instance, for plain similarity over labeled transition systems, the inclusion order $\subseteq$ induces the classic simulation relation while the reversed inclusion $\supseteq$ induces the opposite "simulated by" relation: the first one is right-stable while the second is not.

All these general results arose when trying to integrate two new simulation-like notions as coalgebraic simulations definable by a stable order, so that we could obtain for free all the good properties that have been proved in [6] for this class of relations.

The first new simulation notion is that of covariant-contravariant simulations, where the alphabet of actions Act is partitioned into three disjoint sets Act^l, Act^r, and Act^{bi}. The intention is for the simulation to treat the actions in Act^l like in the ordinary case, to interchange the role of the related processes for those actions in Act^r, and to impose a symmetric condition like that defining bisimulation for the actions in Act^{bi}.

The second notion, conformance simulations, captures the conformance relations [9,12] that several authors introduced in order to formalize the notion of possible implementations. Like covariant-contravariant simulations, they can be defined as coalgebraic simulations for some stable order which is not right-stable neither left-stable. We show that the good properties of these two classes of orders are preserved in those orders that can be seen as a kind of composition of right-stable and left-stable orders. We use this fact to derive the stability of the orders defining both covariant-contravariant and conformance simulations.

2 Coalgebraic Simulations and Stability

Given a category $\mathbb{C}$ and an endofunctor F in $\mathbb{C}$, an F-coalgebra, or just a coalgebra, consists of an object $X \in \mathbb{C}$ together with a morphism $c : X \longrightarrow FX$. We often call X the state space and c the transition or coalgebra structure.

An arbitrary endofunctor F : **Sets** $\longrightarrow$ **Sets** can be lifted to a functor in the category **Rel** over **Sets** $\times$ **Sets** of relations, $\mathrm{Rel}(F)$: **Rel** $\longrightarrow$ **Rel**. In set-theoretic terms, for a relation $R \subseteq X_1 \times X_2$,

$$\mathrm{Rel}(F)(R) = \{\langle u, v\rangle \in FX_1 \times FX_2 \mid \exists w \in F(R).\, F(r_1)(w) = u, F(r_2)(w) = v\}.$$

A *bisimulation* for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ which is "closed under c and d":

$$\text{if } (x, y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)(R),$$

where the r_i are the projections of R into X and Y. Sometimes we shall use the term F-bisimulation to emphasize the functor we are working with.

Bisimulations can also be characterized by means of spans, using the general categorical definition by Aczel and Mendler [1]:

$$\begin{array}{ccccc} X & \xleftarrow{r_1} & R & \xrightarrow{r_2} & Y \\ {\scriptstyle c}\downarrow & & {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle d} \\ FX & \xleftarrow{Fr_1} & FR & \xrightarrow{Fr_2} & FY \end{array}$$

R is a bisimulation iff it is the carrier of some coalgebra e making the above diagram commute. Alternatively, bisimulations can also be defined as the $\mathrm{Rel}(F)$-coalgebras in the category **Rel**.

We will also need the general concept of simulation introduced by Hughes and Jacobs [6] using orders on functors. Let F : **Sets** $\longrightarrow$ **Sets** be a functor. *An order on* F is defined by means of a functorial collection of preorders $\sqsubseteq_X \subseteq FX \times FX$ that must be preserved by renaming: for every $f : X \longrightarrow Y$, if $u \sqsubseteq_X u'$ then $Ff(u) \sqsubseteq_Y Ff(u')$.

Given an order $\sqsubseteq$ on F, a $\sqsubseteq$-*simulation* for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ such that

$$\text{if } (x, y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}_{\sqsubseteq}(F)(R),$$

where the lax relation lifting $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ is $\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X$, which can be expanded to

$$\mathrm{Rel}_{\sqsubseteq}(F)(R) = \{(u, v) \mid \exists w \in F(R).\ u \sqsubseteq_X Fr_1(w) \wedge Fr_2(w) \sqsubseteq_Y v\}.$$

Alternatively, $\sqsubseteq$-simulations are just the $\mathrm{Rel}_{\sqsubseteq}(F)$-coalgebras in **Rel**.

Sometimes, when $f : X \longrightarrow Y$ and $A \subseteq X$ we will simply write $f(A)$ for the image $\coprod_f(A)$.

A functor with order $\sqsubseteq$ is *stable* [6] if the relation lifting $\mathrm{Rel}_{\sqsubseteq}(F)$ commutes with substitution, that is, if for every $f : X \longrightarrow Z$ and $g : Y \longrightarrow W$, $\mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R)) = (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$.[1] They also define a stronger condition that we are going to call right-stability.

Definition 1 ([6]). *We will say that a functor F with order $\sqsubseteq$ is* ***right-stable*** *if, for every function $f : X \longrightarrow Y$, we have*[2]

$$(id \times Ff)^{-1} \sqsubseteq_Y \ \subseteq \coprod_{Ff \times id} \sqsubseteq_X . \tag{2}$$

According to [6], condition (2) is equivalent to (a) F being stable and (b) for every relation $R \subseteq X \times Y$,

$$\mathrm{Rel}(F)(R) \circ \sqsubseteq_X \ \subseteq \ \sqsubseteq_Y \circ \mathrm{Rel}(F)(R). \tag{3}$$

Right-stability was introduced by arguing that it is easier to check than plain stability, while being satisfied by nearly all orders discussed in that paper. Surprisingly, one cannot find in [6] a clear explanation of the reason why right-stable orders are easier to analyze. In our opinion, the crucial fact is that from (3) we can immediately conclude that

$$\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X \ = \ \sqsubseteq_Y \circ \mathrm{Rel}(F)(R), \tag{4}$$

so that the coalgebraic simulations for a right-stable order $\sqsubseteq$ can be equivalently defined by means of the asymmetric definition on the right-hand side of equality (4). If the order $\sqsubseteq$ can be used only on one of the sides of the definition, the verification of the

[1] In fact, the inclusion $\subseteq$ always holds.

[2] Again, the other inclusion is always true since $\sqsubseteq$ functorial means that $Ff(u) \sqsubseteq_Y Ff(v)$ if $u \sqsubseteq_X v$.

properties of the induced coalgebraic simulations becomes much easier than when using the original definition.

It was quite surprising to discover that the easiest way to prove the properties of the "simulated by" relations which come from symmetric properties such as composition-preserving (that are also satisfied by the corresponding inverse relations "simulates") is to break that symmetry by considering the asymmetric definition of coalgebraic simulations that only use $\sqsubseteq_Y$; certainly, this is only possible when the defining order $\sqsubseteq$ is right-stable.

Stability is used in [6, Lemma 5.3] to prove that lax relation lifting preserves composition of relations, which is needed to prove [6, Lemma 5.4(2)], the crucial fact that the induced similarity relation is transitive; this need not be the case for the simulation notion defined by an arbitrary order $\sqsubseteq$.

3 On Stability of Simulation and Anti-simulation

Plain simulations between labeled transition systems can be defined as coalgebraic simulations considering the functor $F = \mathcal{P}^A$ (G^A denote the funtor $X \mapsto (G(X))^A$) with the order $\sqsubseteq$ given by $\alpha \sqsubseteq \beta$ for $\alpha, \beta : A \longrightarrow \mathcal{P}X$ iff for all $a \in A$, $\alpha(a) \subseteq \beta(a)$.

Lemma 1. *The order $\sqsubseteq$ defining plain simulations for labeled transition systems is right-stable.*

Corollary 1. *Plain simulations between labeled transition systems can be defined as the $(\sqsubseteq_Y \circ \mathrm{Rel}(F))$-coalgebras.*

It is worth examining the consequences of the removal of $\sqsubseteq_X$ from the original definition of coalgebraic simulations in this particular case. Both $\sqsubseteq_X$ and $\sqsubseteq_Y$ correspond to the inclusion order, but when applied at the right-hand side it means that we can reduce the set of successors of the simulating process q when simulating the execution of a by p. This means that starting from a set $Y' \subseteq Y$ we can obtain an adequate subset $Y'' \subseteq Y'$. Instead, the application of $\sqsubseteq_X$ at the left-hand side allows to enlarge the set of successors of the simulated process p and this produces a set X'' larger than the given X': one could say that we need to consider "new" information not in X', while going from Y' to Y'' just "removes" some known information.

Another interesting point arises from the fact that every use of $\sqsubseteq_X$ at the left-hand side can be "compensated" by removing at Y the added states and this is why Corollary 1 was correct, because we can always avoid the introduction of new successors in the simulated process by simply removing them at the right-hand side. However, the opposite procedure, to compensate the removal of states by adding them at the simulated process side is not always possible, since in general X could be not big enough.

The anti-simulations can be defined as coalgebraic simulations by taking the reversed inclusion order instead of $\subseteq$. It is interesting to note that it is not right-stable as the following counterexample shows. Let $X = \{x\}$ and $Y = \{y_1, y_2\}$ and let $f : X \longrightarrow Y$ be such that $f(x) = y_1$. With these definitions the pair $(Y, X) \in (id \times \mathcal{P}f)^{-1}(\supseteq)$, since $Y \supseteq \{y_1\} = \mathcal{P}f(X)$, but it is obvious that there is no $A \subseteq X$ such that $Y = f(A)$ because f is not surjective.

However, the order defining anti-simulations is stable as a consequence of the following general result.

Lemma 2. *F with an order $\sqsubseteq$ is stable iff it is stable with the order $\sqsubseteq^{op}$.*

Proof. It is shown in [6, Lemma 4.2(4)] that $\mathrm{Rel}_{\sqsubseteq^{op}}(F)(R) = (\mathrm{Rel}_{\sqsubseteq}(F)(R^{op}))^{op}$. Then, on the one hand,

$$\begin{aligned}(Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq^{op}}(F)(R)) &= (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R^{op}))^{op} \\ &= ((Fg \times Ff)^{-1}\mathrm{Rel}_{\sqsubseteq}(F)(R^{op}))^{op},\end{aligned}$$

and on the other hand,

$$\begin{aligned}\mathrm{Rel}_{\sqsubseteq^{op}}(F)((f \times g)^{-1}(R)) &= (\mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R))^{op})^{op} \\ &= (\mathrm{Rel}_{\sqsubseteq}(F)((g \times f)^{-1}(R^{op})))^{op}.\end{aligned}$$

Since $R^{op} \subseteq Y \times X$ is a relation whenever $R \subseteq X \times Y$ is so, and f, g, and R are arbitrary, we have shown that

$$\mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R)) = (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$$

if and only if

$$\mathrm{Rel}_{\sqsubseteq^{op}}(F)((f \times g)^{-1}(R)) = (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq^{op}}(F)(R)),$$

and therefore F is stable for $\sqsubseteq$ iff it is stable for $\sqsubseteq^{op}$. □

Corollary 2. *The order $\sqsubseteq^{op}$ defining anti-simulations for transition systems as coalgebraic simulations is stable.*

One could conclude from the observation above that there is indeed a natural argument supporting plain similarity as a "right" coalgebraic similarity, definable by a right-stable order. This criterion could be adopted to define right coalgebraic simulations, which plain similarity would satisfy while the opposite relation "is simulated by" would not. However, we immediately noticed that we could define "left-stable" orders by interchanging the roles of Ff and id in the definition of right-stable order, obtaining the inverse inclusion in (1).

Definition 2. *We will say that a functor F with order $\sqsubseteq$ is **left-stable** if, for every function $f : X \longrightarrow Y$, we have*

$$(Ff \times id)^{-1}\ \sqsubseteq_Y\ \subseteq\ \bigsqcup_{id \times Ff} \sqsubseteq_X . \tag{5}$$

It is inmediate to check that an order $\sqsubseteq$ is left-stable iff the inverse order $\sqsubseteq^{op}$ is right-stable. Moreover, left-stable orders have the same structural properties that right-stable ones so that, in particular, they are also stable and hence composition-preserving. But in this case it would be the inverse simulations, corresponding to the "is simulated by" notion, that would be natural instead of plain simulations. As a conclusion, we could use right or left-stability as a criterion to choose a natural direction for the simulation order. But the important fact in both cases is that the simplified asymmetric definitions (using either $\sqsubseteq_X$ or $\sqsubseteq_Y$) of coalgebraic simulations are much easier to handle than the symmetric original definition (where both $\sqsubseteq_X$ and $\sqsubseteq_Y$ have to be used).

4 Covariant-Contravariant Simulations and Conformance Simulations

Covariant-contravariant simulations are defined by combining the conditions "to simulate" and "be simulated by", using a partition of the alphabet of actions of the compared labeled transition systems.

Definition 3. *Given $c : X \longrightarrow \mathcal{P}(X)^{Act}$ and $d : Y \longrightarrow \mathcal{P}(Y)^{Act}$ labeled transition systems for the alphabet Act, and $\{Act^r, Act^l, Act^{bi}\}$ a partition of this alphabet, a (Act^r, Act^l)-**simulation** between c and d is a relation $S \subseteq X \times Y$ such that for every $(x, y) \in S$ we have:*

- *for all $a \in Act^r \cup Act^{bi}$ and all $x \xrightarrow{a} x'$ there exists $y \xrightarrow{a} y'$ with $(x', y') \in S$.*
- *for all $a \in Act^l \cup Act^{bi}$, and all $y \xrightarrow{a} y'$ there exists $x \xrightarrow{a} x'$ with $(x', y') \in S$.*

We write $x \, {}_{Act^r}\mathcal{S}_{Act^l} \, y$, and say that x is (Act^r, Act^l)-simulated by y, if and only if there exists some (Act^r, Act^l)-simulation S with $xS\,y$.

A very interesting application of this kind of simulations is related with the definition of adequate simulation notions for input/output (I/O) automata [10]. The classic approach to simulations is based on the definition of semantics for reactive systems, where all the actions of the processes correspond to input actions that the user must trigger. Instead, whenever we have explicit output actions the situation is the opposite: it is the system that produces the actions and the user who is forced to accept the produced output. Then, it is natural to conclude that in the simulation framework we have to dualize the simulation condition when considering output actions, and this is exactly what our anti-simulation relations do.

Covariant-contravariant simulations can be easily obtained as coalgebraic simulations, as the following proposition proves.

Proposition 1. *(Act^r, Act^l)-simulations can be defined as the coalgebraic simulations for the functor $F = \mathcal{P}^{Act}$ with functorial order ${}_{Act^r}\sqsubseteq_{Act^l}$ where, for each set X and $\alpha, \alpha' : Act \longrightarrow \mathcal{P}(X)$, we have $\alpha \, {}_{Act^r}\sqsubseteq_{Act^l} \alpha'$ if:*

- *for all $a \in Act^r \cup Act^{bi}$, $\alpha(a) \subseteq \alpha'(a)$, and*
- *for all $a \in Act^l \cup Act^{bi}$, $\alpha(a) \supseteq \alpha'(a)$.*

Note that in particular we have $\alpha(a) = \alpha'(a)$ for all $a \in Act^{bi}$.

Proof. Intuitively, using the order ${}_{Act^r}\sqsubseteq_{Act^l}$ on the left-hand side of $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ allows us to remove a'-transitions when $a' \in Act^l$, whereas using it on the right-hand side of $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ allows us to remove a-transitions when $a \in Act^r$.

Let us suppose that we have a classic covariant-contravariant simulation ${}_{Act^r}\mathcal{S}_{Act^l}$ between labeled transition systems $c : P \longrightarrow \mathcal{P}(P)^{Act}$ and $d : Q \longrightarrow \mathcal{P}(Q)^{Act}$ defined by $c(p)(a) = \{p' \mid p \xrightarrow{a} p'\}$ and $d(q)(a) = \{q' \mid q \xrightarrow{a} q'\}$. We must show that if $p \, {}_{Act^r}\mathcal{S}_{Act^l} \, q$ then there exist p^* and q^* such that

$$c(p) \, {}_{Act^r}\sqsubseteq_{Act^l} \, p^* \mathrm{Rel}(\mathcal{P}^{Act})({}_{Act^r}\mathcal{S}_{Act^l}) q^* \, {}_{Act^r}\sqsubseteq_{Act^l} \, d(q). \tag{6}$$

We define p^* and q^* as follows:

- p^* has the same transitions as $c(p)$, except for those transitions $p \xrightarrow{a'} p'$ with $a' \in Act^l$ such that there is no q' with $q \xrightarrow{a'} q'$ and $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$.
- q^* has the same transitions as $d(q)$, except for those transitions $q \xrightarrow{a} q'$ with $a \in Act^r$ such that there is no p' with $p \xrightarrow{a} p'$ and $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$.

It is immediate from these definitions that $c(p) \; {}_{Act^r}\sqsubseteq_{Act^l} p^*$ and $q^* \; {}_{Act^r}\sqsubseteq_{Act^l} d(q)$, so we are left with checking that $p^*\mathrm{Rel}(\mathcal{P}^{Act})q^*$.

Let $p' \in p^*(a)$ with $a \in Act^r$. By construction of p^*, since we have not dropped any a-transitions from p^*, $p \xrightarrow{a} p'$. Using the fact that ${}_{Act^r}\mathcal{S}_{Act^l}$ is a classic covariant-contravariant simulation, there exists q' such that $q \xrightarrow{a} q'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$, and, again by construction, $q' \in q^*(a)$ because there is some $p \xrightarrow{a} p'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Similarly, if $p' \in p^*(a)$ with $a' \in Act^l$, by construction of p^* there must exist some q' such that $q \xrightarrow{a'} q'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Again, since we have not removed any a'-transitions from $d(q)$ in q^*, it must be true that $q' \in q^*(a)$. Finally, if $p' \in p^*(a)$ with $a \in Act^{bi}$ we have that $p \xrightarrow{a} p'$ and hence there exists q' such that $q \xrightarrow{a} q'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$, but also $q' \in q^*(a)$.

The argument that shows that for every $q' \in q^*(a)$ there exists some $p' \in p^*(a)$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$ is analogous.

We show now the other implication, that a coalgebraic covariant-contravariant simulation is a classic one. In this case we start from coalgebras c and d that satisfy relation (6) whenever $p \; {}_{Act^r}\mathcal{S}_{Act^l} \; q$.

If $p \xrightarrow{a} p'$ for $a \in Act^r$, then $p' \in p^*(a)$ because $c(p) \; {}_{Act^r}\sqsubseteq_{Act^l} p^*$ and, since $p^*\mathrm{Rel}(\mathcal{P}^{Act})({}_{Act^r}\mathcal{S}_{Act^l})q^*$, there is some $q' \in q^*(a)$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Again, the definition of ${}_{Act^r}\sqsubseteq_{Act^l}$ ensures that $q^*(a) \subseteq d(q)(a)$ and hence $q \xrightarrow{a} q'$ as required. Similarly, if $q \xrightarrow{a'} q'$ for $a' \in Act^l$, then $q' \in q^*(a)$ because $q^* \; {}_{Act^r}\sqsubseteq_{Act^l} d(q)$ and thus, as in the previous case, there exists $p' \in p^*(a)$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$ and $p \xrightarrow{a'} p'$. Finally if $p \xrightarrow{a} p'$ for $a \in Act^{bi}$ (resp. $q \xrightarrow{a} q'$), again by the definition of ${}_{Act^r}\sqsubseteq_{Act^l}$ we have $p' \in p^*(a)$ (resp. $q' \in q^*(a)$) and, from $p^*\mathrm{Rel}(\mathcal{P}^{Act})({}_{Act^r}\mathcal{S}_{Act^l})q^*$, it follows that there exists $q' \in q^*(a)$ (resp. $p' \in p^*(a)$) such that $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$; by the definition of ${}_{Act^r}\sqsubseteq_{Act^l}$, $q \xrightarrow{a} q'$ (resp. $p \xrightarrow{a} p'$). □

The other new kind of simulations in which we are interested is that of conformance simulations, where the conformance relation in [9,12] meets the simulation world in a nice way. In the definition below we will write $p \xrightarrow{a}$ if $p \xrightarrow{a} p'$ for some p'.

Definition 4. *Given $c : X \longrightarrow \mathcal{P}(X)^A$ and $d : Y \longrightarrow \mathcal{P}(Y)^A$ two labeled transition systems for the alphabet A, a **conformance simulation** between them is a relation $R \subseteq X \times Y$ such that whenever pRq, then:*

- *For all $a \in A$, if $p \xrightarrow{a}$ we must also have $q \xrightarrow{a}$ (this means, using the usual notation for process algebras, that $I(p) \subseteq I(q)$).*
- *For all $a \in A$ such that $q \xrightarrow{a} q'$ and $p \xrightarrow{a}$, there exists some p' with $p \xrightarrow{a} p'$ and $p'Rq'$.*

Conformance simulations allow the extension of the set of actions offered by a process, so that in particular we will have $a < a + b$, but they also consider that a process can be "improved" by reducing the nondeterminism in it, so that $ap + aq < ap$. In this way we have again a kind of covariant-contravariant simulation, not driven by the alphabet of actions executed by the processes but by their nondeterminism.

Once again, conformance simulations can be defined as coalgebraic simulations taking the adequate order on the functor defining labeled transition systems.

Proposition 2. *Conformance simulations can be obtained as the coalgebraic simulations for the order $\sqsubseteq^{Conf}$ on the functor $\mathcal{P}^A$, where for any set X we have $u \sqsubseteq_X^{Conf} v$ if for every $u, v : A \longrightarrow \mathcal{P}X$ and $a \in A$:*

- *either $u(a) = \emptyset$, or*
- *$u(a) \supseteq v(a)$ and $v(a) \neq \emptyset$.*

Proof. Let us first prove that $\sqsubseteq_X^{Conf}$ is indeed an order. It is clear that the only not immediate property is transitivity. To check it, let us take $u \sqsubseteq_X^{Conf} v \sqsubseteq_Y^{Conf} w$: if $u(a) = \emptyset$ we are done; otherwise, we have $u(a) \supseteq v(a)$ and $v(a) \neq \emptyset$, so that we also have $v(a) \supseteq w(a)$ and $w(a) \neq \emptyset$, obtaining $u(a) \supseteq w(a)$ and $w(a) \neq \emptyset$.

Now, we can interpret that using the order $\sqsubseteq^{Conf}$ on the left-hand side of $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ allows us to remove all a-transitions except for the last one, whereas using it on the right-hand side allows us to remove all b-transitions for $b \in B$, where B is any set of actions. But again, as in the proof of Proposition 1, we can compensate these additions with the corresponding removals at the other side and the proof follows in an analogous way. □

Next we check that the order ${}_{Act^r}\sqsubseteq_{Act^l}$ defining covariant-contravariant simulations is stable.

Lemma 3. *Given a partition $\{Act^r, Act^l, Act^{bi}\}$ of Act the order ${}_{Act^r}\sqsubseteq_{Act^l}$ for the functor $\mathcal{P}^{Act}$ defining covariant-contravariant simulations for transition systems is stable.*

Proof. It is clear that the order ${}_{Act^r}\sqsubseteq_{Act^l}$ can be obtained as the product of a family of orders $\sqsubseteq^a$ for the functor $\mathcal{P}$, with $a \in Act$. This is indeed the case taking $\sqsubseteq_X^a = \subseteq_X$ for $a \in Act^r$, $\sqsubseteq_X^a = \supseteq_X$ for $a \in Act^l$ and $\sqsubseteq_X^a = =_X$ for $a \in Act^{bi}$. Then it is easy to see that to obtain that ${}_{Act^r}\sqsubseteq_{Act^l}$ is stable it is enough to prove that each of the orders $\sqsubseteq^a$ is stable.

This latter requirement is straightforward because, for $a \in Act^r$, $\sqsubseteq^a$ is right-stable; for $a \in Act^l$ the order $\sqsubseteq^a$ is left-stable; and for $a \in Act^{bi}$, $\sqsubseteq^a$ is the equality relation, which is both right and left-stable, for every functor F. □

Certainly, the order defining covariant-contravariant simulations is not right-stable nor left-stable, but in the proof above we have used the power of these two properties thanks to the fact that the order ${}_{Act^r}\sqsubseteq_{Act^l}$ can be factorised as the product of a family of orders that are either right-stable or left-stable. Then we can obtain the following sequence of general definitions and results, from which Lemma 3 could be obtained as a simple particular case.[3]

[3] Instead of removing the above, we have preferred to maintain the sequence of results in the order in which we got them, starting with our motivating example.

Definition 5. *We say that an order $\sqsubseteq$ on a functor F^A is **action-distributive** if there is a family of orders $\sqsubseteq^a$ on F such that*

$$f \sqsubseteq g \iff f(a) \sqsubseteq^a g(a) \text{ for all } a \in A.$$

Whenever $\sqsubseteq$ can be distributed in this way we will write $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$.

Definition 6. *We say that an action-distributive order $\sqsubseteq$ on F^A is **side stable** if for the decomposition $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$ we have that each order $\sqsubseteq^a$ is either right-stable or left-stable.*

By separating the right-stable and the left-stable components we obtain $\sqsubseteq = \sqsubseteq^l \times \sqsubseteq^r$, where A^r (resp. A^l) collects the set of arguments[4] $a \in A$ with $\sqsubseteq^a$ right-stable (resp. left-stable). We extend $\sqsubseteq^l$ and $\sqsubseteq^r$ to obtain a pair of orders on F^A, $\sqsubseteq^{\bar{l}}$ and $\sqsubseteq^{\bar{r}}$, defined by:

- *$f \sqsubseteq^{\bar{r}} g$ iff $f(a) \sqsubseteq^a g(a)$ for all $a \in A^r$ and $f(a) = g(a)$ for all $a \in A^l$.*
- *$f \sqsubseteq^{\bar{l}} g$ iff $f(a) \sqsubseteq^a g(a)$ for all $a \in A^l$ and $f(a) = g(a)$ for all $a \in A^r$.*

Proposition 3. *The order $\sqsubseteq^{\bar{l}}$ is left-stable, while $\sqsubseteq^{\bar{r}}$ is right-stable. We have $\sqsubseteq = (\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}}) = (\sqsubseteq^{\bar{r}} \circ \sqsubseteq^{\bar{l}})$, and therefore we also have $\sqsubseteq = (\sqsubseteq^{\bar{l}} \cup \sqsubseteq^{\bar{r}})^*$.*

Proposition 4. *For any side stable order $\sqsubseteq$ on F^A, if we have a decomposition $\sqsubseteq = \sqsubseteq^l \times \sqsubseteq^r$ based on a partition of A into a set of right-stable components A^r and another set of left-stable components A^l, then we can obtain the coalgebraic simulations for $\sqsubseteq$ as the $(\sqsubseteq^{\bar{r}}_Y \circ \mathrm{Rel}(F) \circ \sqsubseteq^{\bar{l}}_X)$-coalgebras.*

Proof. By definition, $\mathrm{Rel}_{\sqsubseteq}(F)(R) = \sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X$. Since $\sqsubseteq = (\sqsubseteq^{\bar{r}} \circ \sqsubseteq^{\bar{l}}) = (\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}})$, we have:

$$\begin{aligned}
\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X &= (\sqsubseteq^{\bar{l}}_Y \circ \sqsubseteq^{\bar{r}}_Y) \circ \mathrm{Rel}(F)(R) \circ (\sqsubseteq^{\bar{r}}_X \circ \sqsubseteq^{\bar{l}}_X) \\
&= \sqsubseteq^{\bar{l}}_Y \circ (\sqsubseteq^{\bar{r}}_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{r}}_X) \circ \sqsubseteq^{\bar{l}}_X \\
&= (\sqsubseteq^{\bar{l}}_Y \circ \sqsubseteq^{\bar{r}}_Y) \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{l}}_X \qquad \text{(by right-stability of } \sqsubseteq^{\bar{r}}) \\
&= \sqsubseteq^{\bar{r}}_Y \circ (\sqsubseteq^{\bar{l}}_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{l}}_X) \qquad \text{(since } \sqsubseteq^{\bar{r}} \text{ and } \sqsubseteq^{\bar{l}} \text{ commute)} \\
&= \sqsubseteq^{\bar{r}}_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{l}}_X \qquad \text{(by left-stability of } \sqsubseteq^{\bar{l}})
\end{aligned}$$

□

The characterization above still requires the use of the order on both sides of the $\mathrm{Rel}(F)(R)$ operator. However, the fact that $\sqsubseteq^{\bar{r}}_Y$ (resp. $\sqsubseteq^{\bar{l}}_X$) is right-stable (resp. left-stable) makes the application of this decomposition as simple as when coping with either a right or left-stable order.

Proposition 5. *If $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$ and $\sqsubseteq^a$ is stable for all $a \in A$, then $\sqsubseteq$ is stable.*

[4] We have assumed here a partition $\{A^l, A^r\}$ of the set A into two sets of right-stable and left-stable components. Obviously, if there were some arguments $a \in A$ on which $\sqsubseteq^a$ is both right-stable and left-stable then the decomposition would not be unique, but the result would be valid for any such decomposition.

Proof. The result follows from the following chain of implications:

$$\begin{array}{ll} & (u,v) \in (Ff \times Fg)^{-1}\mathrm{Rel}_{\sqsubseteq}(F)(R) \\ \iff & Ff(u) \sqsubseteq z'\mathrm{Rel}(F)(R)w' \sqsubseteq Fg(v) \\ \iff & Ff(u)(a) \sqsubseteq^a z'(a)\mathrm{Rel}(F^a)(R)w'(a) \sqsubseteq^a Fg(v)(a), \text{ for all } a \\ \iff & (u(a),v(a)) \in (Ff \times Fg)^{-1}\mathrm{Rel}_{\sqsubseteq^a}(F)(R), \text{ for all } a \\ \implies & (u(a),v(a)) \in \mathrm{Rel}_{\sqsubseteq^a}(F)((f \times g)^{-1}R), \text{ for all } a \\ \iff & u(a) \sqsubseteq^a x'(a)\mathrm{Rel}(F)((f \times g)^{-1}R)y'(a) \sqsubseteq^a v(a), \text{ for all } a \\ \iff & (u,v) \in \mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}R) \end{array}$$

□

Corollary 3. *Any side stable order is stable.*

Corollary 4. *The order ${}_{Act^r}\sqsubseteq_{Act^l}$ defining covariant-contravariant simulations is side stable and therefore it is stable too.*

Next we consider the case of conformance simulations, for which we can obtain similar results to those proved for covariant-contravariant simulations.

Lemma 4. *The order $\sqsubseteq^{Conf}$ defining conformance simulations for transition systems is stable.*

Proof. Let $R \subseteq Z \times W$ be a relation and $f : X \longrightarrow Z$, $g : Y \longrightarrow W$ arbitrary functions. If $(u,v) \in (\mathcal{P}^A f \times \mathcal{P}^A g)^{-1}(\mathrm{Rel}_{\sqsubseteq^{Conf}}(\mathcal{P}^A)(R))$, then there exist z and w such that

$$\mathcal{P}^A f(u) \sqsubseteq^{Conf} z \ \mathrm{Rel}(\mathcal{P}^A)(R) \ w \sqsubseteq^{Conf} \mathcal{P}^A g(v). \tag{7}$$

We have to show that $(u,v) \in \mathrm{Rel}_{\sqsubseteq^{Conf}}(\mathcal{P}^A)((f \times g)^{-1}(R))$, that is, there exist x and y such that

$$u \sqsubseteq^{Conf} x \ \mathrm{Rel}(\mathcal{P}^A)((f \times g)^{-1}(R)) \ y \sqsubseteq^{Conf} v.$$

Let us define $x : A \longrightarrow \mathcal{P}(X)$ by $x(a) = u(a) \cap f^{-1}(z(a))$ and $y : A \longrightarrow \mathcal{P}(Y)$ by $y(a) = g^{-1}(w(a))$. Then we have:

1. $u \sqsubseteq^{Conf} x$.
   If $u(a) = \emptyset$, there is nothing to prove. Otherwise, since $\mathcal{P}^A f(u) \sqsubseteq^{Conf} z$ and $f(u(a)) \neq \emptyset$, we have $f(u(a)) \supseteq z(a) \neq \emptyset$ and hence $u(a) \supseteq u(a) \cap f^{-1}(z(a)) = x(a) \neq \emptyset$.
2. $y \sqsubseteq^{Conf} v$.
   If $w(a) = \emptyset$, then $y(a) = g^{-1}(w(a)) = \emptyset$. Otherwise, since $w \sqsubseteq^{Conf} \mathcal{P}^A g(v)$, we have $w(a) \supseteq g(v(a)) \neq \emptyset$, so that $v(a) \neq \emptyset$ and $y(a) = g^{-1}(w(a)) \supseteq g^{-1}(g(v(a))) \supseteq v(a)$.
3. $x \ \mathrm{Rel}(\mathcal{P}^A)((f \times g)^{-1}(R)) \ y$.
   For every $a \in A$ we need to show that $x(a) \ \mathrm{Rel}(\mathcal{P})((f \times g)^{-1}(R)) \ y(a)$, which means:
   (a) for every $p \in x(a)$ there exists $q \in y(a)$ such that $p \ (f \times g)^{-1}(R) \ q$, that is, $f(p)Rg(q)$; and
   (b) for every $q \in y(a)$ there exists $p \in x(a)$ such that $p \ (f \times g)^{-1}(R) \ q$, that is, $f(p)Rg(q)$.
   In the first case, let $p \in x(a)$; by definition of x, $f(p) \in z(a)$. Now, from $z \ \mathrm{Rel}(\mathcal{P}^A)(R) \ w$ we obtain that for each $p' \in z(a)$ there exists $q' \in w(a)$ such that $p'Rq'$. Then, for $f(p) \in z(a)$ there exists $q' \in w(a)$ with $f(p)Rq'$; and by definition of y, there exists $q \in y(a)$ with $q' = g(q)$ as required.
   In the second case, let $q \in y(a)$ so that $g(q) \in w(a)$. Again, from $z \ \mathrm{Rel}(\mathcal{P}^A)(R) \ w$ it follows that there is $p' \in z(a)$ with $p'Rg(q)$. Now, $f(u(a)) \supseteq z(a)$ because $u \sqsubseteq^{Conf} z$, so there exists $p \in u(a) \cap f^{-1}(z(a))$ with $f(p) = p'$, as required. □

As in the case of covariant-contravariant simulations, conformance simulations cannot be defined as coalgebraic simulations using neither a right-stable order nor a left-stable order. But we can find in the arguments above the basis for a decomposition of the involved order $\sqsubseteq^{Conf}$, according to the two cases in its definition. Once again $\sqsubseteq^{Conf}$ is an action-distributive order on $\mathcal{P}^A$, but in order to obtain the adequate decomposition of $\sqsubseteq^{Conf}$ now we also need to decompose the component orders $\sqsubseteq^a$.

Definition 7. *We define the conformance orders $\sqsubseteq^{C\neg\emptyset}$, $\sqsubseteq^{C\emptyset}$, and $\sqsubseteq^{C}$ on the functor $\mathcal{P}$ by:*

- $x_1 \sqsubseteq^{C\emptyset} x_2$ *if* $x_1 = \emptyset$ *or* $x_1 = x_2$.
- $x_1 \sqsubseteq^{C\neg\emptyset} x_2$ *if* $x_1 \supseteq x_2$ *and* $x_2 \neq \emptyset$, *or* $x_1 = x_2$.
- $x_1 \sqsubseteq^{C} x_2$ *if* $x_1 \sqsubseteq^{C\neg\emptyset} x_2$ *or* $x_1 \sqsubseteq^{C\emptyset} x_2$.

Proposition 6. *The two relations $\sqsubseteq^{C\emptyset}$ and $\sqsubseteq^{C\neg\emptyset}$ commute with each other:*

$$(\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset}) = (\sqsubseteq^{C\neg\emptyset} \circ \sqsubseteq^{C\emptyset}),$$

from where it follows that $(\sqsubseteq^{C\emptyset} \cup \sqsubseteq^{C\neg\emptyset})^* = (\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset}) = (\sqsubseteq^{C\neg\emptyset} \circ \sqsubseteq^{C\emptyset})$. We also have $\sqsubseteq^{C} = (\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset})$, from where we conclude that $\sqsubseteq^{C}$ is indeed an order relation.

Proof. Let u $(\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset})$ v: there is some w such that $u \sqsubseteq^{C\neg\emptyset} w$ and $w \sqsubseteq^{C\emptyset} v$. We need to find w' such that $u \sqsubseteq^{C\emptyset} w'$ and $w' \sqsubseteq^{C\neg\emptyset} v$. If $w = \emptyset$ then it must be $u = \emptyset$ too, and we can take $w' = v$; otherwise, it must be $v = w$ and we can take $w' = u$. The other inclusion is similar. □

Corollary 5. *The order $\sqsubseteq^{Conf}$ defining conformance simulations can be decomposed into $\prod_{a\in A} \sqsubseteq^a$ where, for each $a \in A$, we have $\sqsubseteq^a = \sqsubseteq^C$ as defined above. Then, $\sqsubseteq^{Conf} = \prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset} \cup \sqsubseteq^{a,\emptyset})^* = \prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset}) \circ \prod_{a\in A}(\sqsubseteq^{a,\emptyset}) = \prod_{a\in A}(\sqsubseteq^{a,\emptyset}) \circ \prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset})$, so that we obtain $\sqsubseteq^{Conf}$ as the composition of a right-stable order and a left-stable order that commute with each other.*

Proposition 7. *For any pair of right (resp. left)-stable orders $\sqsubseteq^1$, $\sqsubseteq^2$ on F, their composition also defines a right (resp. left)-stable order on F.*

Proof. Given $f : X \longrightarrow Y$ we must show that

$$(id \times Ff)^{-1}(\sqsubseteq^1_Y \circ \sqsubseteq^2_Y) \subseteq \coprod_{(Ff\times id)} (\sqsubseteq^1_X \circ \sqsubseteq^2_X).$$

Let us assume that $(y, x) \in (id \times Ff)^{-1}(\sqsubseteq^1_Y \circ \sqsubseteq^2_Y)$, that is, y $(\sqsubseteq^1 \circ \sqsubseteq^2)$ $y' = Ff(x)$; then, there exists $y'' \in FY$ such that $y \sqsubseteq^2_Y y''$ and $y'' \sqsubseteq^1_Y y'$. Graphically,

$$\begin{array}{ccccc} y & \sqsubseteq^2_Y & y'' & \sqsubseteq^1_Y & y' \\ & & & & \uparrow Ff \\ & & & & x \end{array} \tag{8}$$

Since $\sqsubseteq^1_Y$ is right-stable we have that $(id \times Ff)^{-1}\ \sqsubseteq^1_Y \subseteq \coprod_{(Ff\times id)} \sqsubseteq^1_X$. Hence, there exists $x'' \in FX$ such that $Ff(x'') = y''$ and $x''\ \sqsubseteq^1_X\ x$, thus turning diagram (8) into the following:

$$\begin{array}{ccc} y & \sqsubseteq^2_Y & y'' \\ & & \uparrow{\scriptstyle Ff} \\ & & x'' \quad \sqsubseteq^1_X \quad x \end{array} \tag{9}$$

Now, we can apply right-stability of $\sqsubseteq^2$: since we have $(y, x'') \in (id \times Ff)^{-1}\ \sqsubseteq^2_Y \subseteq \coprod_{(Ff\times id)} \sqsubseteq^2_X$, there exists $x' \in FX$ such that $Ff(x') = y$ and $x'\ \sqsubseteq^2_X\ x''$. Thus, diagram (9) becomes

$$\begin{array}{ccccccc} y & & & & & & \\ \uparrow{\scriptstyle Ff} & & & & & & \\ x' & \sqsubseteq^2_X & x'' & \sqsubseteq^1_X & x & & \end{array} \tag{10}$$

which means that there exist $x', x'' \in FX$ such that $Ff(x') = y$, $x'\ \sqsubseteq^2_X\ x''$ and $x''\ \sqsubseteq^1_X\ x$, or equivalently, that $(y, x) \in \coprod_{(Ff\times id)}(\sqsubseteq^1_X \circ \sqsubseteq^2_X)$, as we had to prove. □

Proposition 8. *If $\sqsubseteq^r$ is a right-stable order on F and $\sqsubseteq^l$ is a left-stable order on F that commute with each other, then their composition defines a stable order on F. Moreover, the coalgebraic simulations for the order $\sqsubseteq\ =\ \sqsubseteq^r \circ \sqsubseteq^l$ can be equivalently defined as the $(\sqsubseteq^r \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^l)$-coalgebras.*

Proof. Let $R \subseteq Z \times W$ be a relation, $f : X \longrightarrow Z$ and $g : Y \longrightarrow W$ arbitrary functions, and $\sqsubseteq\ =\ \sqsubseteq^r \circ \sqsubseteq^l$. Let us suppose that $(u, v) \in (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$. Then, since $\sqsubseteq^r$ and $\sqsubseteq^l$ commute with each other, using Proposition 4, there exist z', w' such that

$$Ff(u)\ \sqsubseteq^l_Z\ z'\ \mathrm{Rel}(F)(R)\ w'\ \sqsubseteq^r_W\ Fg(v). \tag{11}$$

If we write z for $Ff(u)$ and w for $Fg(v)$, then equation (11) is equivalent to

$$\begin{array}{ccccccccc} z & \sqsubseteq^l_Z & z' & \mathrm{Rel}(F)(R) & w' & \sqsubseteq^r_W & w \\ {\scriptstyle Ff}\uparrow & & & & & & \uparrow{\scriptstyle Fg} \\ u & & & & & & v \end{array} \tag{12}$$

and we have to show that $(u, v) \in \mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R))$, that is, that there exist x and y such that

$$u\ \sqsubseteq^l_X\ x\ \mathrm{Rel}(F)((f \times g)^{-1}(R))\ y\ \sqsubseteq^r_Y\ v.$$

Using that $\sqsubseteq^r$ is right-stable on the rhs of equation (11), we get $(w', v) \in (id \times Fg)^{-1}\ \sqsubseteq^r_W \subseteq \coprod_{(Fg\times id)} \sqsubseteq^r_Y$, so that there is some $y \in FY$ such that $Fg(y) = w'$, with $y\ \sqsubseteq^r_Y\ v$. Graphically, diagram (12) becomes

$$\begin{array}{ccccccc} z & \sqsubseteq^l_Z & z' & \mathrm{Rel}(F)(R) & w' & & \\ {\scriptstyle Ff}\uparrow & & & & \uparrow{\scriptstyle Fg} & & \\ u & & & & y & \sqsubseteq^r_Y & v \end{array} \tag{13}$$

Analogously, applying the left-stability of order $\sqsubseteq_Z^l$ we get that there is some $x \in FX$ with $Ff(x) = z'$ such that $u \sqsubseteq_X^l x$. Or graphically,

$$\begin{array}{ccccccc} & & z' & \mathrm{Rel}(F)(R) & w' & & \\ & & {\scriptstyle Ff}\uparrow & & \uparrow{\scriptstyle Fg} & & \\ u & \sqsubseteq_X^l & x & & y & \sqsubseteq_Y^r & v \end{array} \tag{14}$$

But diagram (14) is just what we had to prove, since we have found x, y such that $(x, y) \in (Ff \times Fg)^{-1}(\mathrm{Rel}(F)(R)) = \mathrm{Rel}(F)((f \times g)^{-1}(R))$ with $u \sqsubseteq_X^l x$, $y \sqsubseteq_Y^r v$ or, in other words, $(u, v) \in \mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R))$. □

In particular, for our running example of conformance simulations we obtain the corresponding factorization of the definition of coalgebraic simulations for the order $\sqsubseteq^{Conf}$:

Corollary 6. *Coalgebraic simulations for the conformance order $\sqsubseteq^{Conf}$ can be equivalently defined as the $(\prod_{a\in A}(\sqsubseteq_Y^{a,\neg\emptyset}) \circ \mathrm{Rel}(F)(R) \circ \prod_{a\in A}(\sqsubseteq_X^{a,\emptyset}))$-coalgebras.*

5 Conclusion

We have presented in this paper two new simulation orders induced by two criteria that capture the difference between input and output actions and the implementation notions that are formalized by the conformance relations.

In order to apply the general theory of coalgebraic simulations to them, we identified the corresponding orders on the functor defining labeled transition systems. However, it was not immediate to prove that the obtained orders had the desired good properties since the usual way to do it, namely, by establishing stability as a consequence of a stronger property that we have called right-stability, is not applicable in this case.

Trying to adapt that property to our situation we have discovered several interesting consequences. We highlight the fact that right-stability is an assymetric property which has proved to be very useful for the study of a "reversible" concept such as that of relation, since it is clear that any structural result on the theory of relations should remain true when we reverse the relations, simply "observing" them "from the other side". Two consequences of that assymetric approach followed: first we noticed that we could use it to point the simulation orders in some natural way; secondly we also noticed that by dualizing the right-stability condition we could obtain left-stability.

But the crucial result in order to be able to manage more complicated simulation notions, as proved to be the case for our new covariant-contravariant simulations and the conformance simulations, was the discovery of the fact that both of them could be factorized into the composition of a right-stable and a left-stable component. Exploiting this decomposition we have been able to easily adapt all the techniques that had proved to be very useful for the case of right-stable orders.

We plan to expand our work here in two different directions. The first one is concerned with the two new simulated notions introduced in this paper: once we know that they can be defined as stable coalgebraic simulations and therefore have all the desired basic properties of simulations, we will continue with their study by integrating them

into our unified presentation of the semantics for processes [4]. Hence we expect to obtain, in particular, a clear relation between conformance similarity and the classic similarity orders as well as an algebraic characterization for the new semantics. In addition, we plan to continue with our study of stability, which has proved to be a crucial property in order to understand the notion of coalgebraic simulation, thus making it possible to apply the theory to other examples like those studied in this paper.

References

1. Aczel, P., Mendler, N.P.: A final coalgebra theorem. In: Dybjer, P., Pitts, A.M., Pitt, D.H., Poigné, A., Rydeheard, D.E. (eds.) Category Theory and Computer Science. LNCS, vol. 389, pp. 357–365. Springer, Heidelberg (1989)
2. Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. J. ACM 42(1), 232–268 (1995)
3. de Frutos-Escrig, D., Gregorio-Rodríguez, C.: Universal coinductive characterisations of process semantics. In: Ausiello, G., Karhumäki, J., Mauri, G., Ong, C.-H.L. (eds.) IFIP TCS. IFIP, vol. 273, pp. 397–412. Springer, Heidelberg (2008)
4. de Frutos Escrig, D., Gregorio-Rodríguez, C., Palomino, M.: On the unification of semantics for processes: observational semantics. In: Nielsen, M., Kucera, A., Miltersen, P.B., Palamidessi, C., Tuma, P., Valencia, F.D. (eds.) SOFSEM 2009. LNCS, vol. 5404, pp. 279–290. Springer, Heidelberg (2009)
5. Groote, J.F., Vaandrager, F.W.: Structured operational semantics and bisimulation as a congruence. Inf. Comput. 100(2), 202–260 (1992)
6. Hughes, J., Jacobs, B.: Simulations in coalgebra. TCS 327(1-2), 71–108 (2004)
7. Jacobs, B., Hughes, J.: Simulations in coalgebra. In: Gumm, H.P. (ed.) CMCS 2003: 6th International Workshop on Coalgebraic Methods in Computer Science, vol. 82 (2003)
8. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Inf. Comput. 94(1), 1–28 (1991)
9. Leduc, G.: A framework based on implementation relations for implementing LOTOS specifications. Computer Networks and ISDN Systems 25(1), 23–41 (1992)
10. Lynch, N.A., Tuttle, M.R.: Hierarchical correctness proofs for distributed algorithms. In: Sixth Annual ACM Symposium on Principles of Distributed Computing, pp. 137–151 (1987)
11. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
12. Tretmans, J.: Conformance testing with labelled transition systems: Implementation relations and test generation. Computer Networks and ISDN Systems 29(1), 49–79 (1996)
13. van Glabbeek, R.J.: The linear time-branching time spectrum I: The semantics of concrete, sequential processes. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of process algebra, pp. 3–99 (2001)

Multiset Bisimulations as a Common Framework for Ordinary and Probabilistic Bisimulations*

David de Frutos Escrig, Miguel Palomino, and Ignacio Fábregas

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid
{defrutos,miguelpt}@sip.ucm.es, fabregas@fdi.ucm.es

Abstract. Our concrete objective is to present both ordinary bisimulations and probabilistic bisimulations in a common coalgebraic framework based on multiset bisimulations. For that we show how to relate the underlying powerset and probabilistic distributions functors with the multiset functor by means of adequate natural transformations. This leads us to the general topic that we investigate in the paper: a natural transformation from a functor F to another G transforms F-bisimulations into G-bisimulations but, in general, it is not possible to express G-bisimulations in terms of F-bisimulations. However, they can be characterized by considering Hughes and Jacobs' notion of simulation, taking as the order on the functor F the equivalence induced by the epi-mono decomposition of the natural transformation relating F and G. We also consider the case of alternating probabilistic systems where non-deterministic and probabilistic choices are mixed, although only in a partial way, and extend all these results to categorical simulations.

1 Introduction

Bisimulations are the adequate way to capture behavioural indistinguishability of states of systems. Ordinary bisimulations were introduced [11] to cope with labelled transition systems and other similar models and have been used to define the formal observational semantics of many popular languages and formalisms, such as CCS. Bisimilarity is also the natural way to express equivalence of states in any system described by means of a coalgebra over an arbitrary functor F. The general categorical definition can be presented in a more concrete way for the class of polynomial functors, that are defined by means of a simple signature of constructors and whose properties, including the definition of relation lifting, can be studied by means of structural induction. In particular, the powerset constructor is one of them, and therefore the class of labelled transition systems can be studied as a simple and illustrative example of the categorical framework.

The simplicity and richness of the theory of bisimulations made it interesting to define several extensions in which the structure on the set of labels of

* Research supported by the Spanish projects DESAFIOS TIN2006-15660-C02-01, WEST TIN2006-15578-C02-01 and PROMESAS S-0505/TIC/0407.

K. Suzuki et al. (Eds.): FORTE 2008, LNCS 5048, pp. 283–298, 2008.

the considered systems was taken into account, instead of the plain approach made by simple (strong) bisimulations. For instance, weak bisimulation takes into account the existence of non-observable actions, while timed and probabilistic bisimulation introduce timed or probabilistic features. In particular, the original definition of probabilistic bisimulation for probabilistic transition systems had to capture the fact that one should be able to accumulate the probabilities of several transitions arriving at equivalent (bisimilar) states in order to simulate some transition or, conversely, that one should be able to distribute the probability of a transition among several others connecting the same states.

The classical definition by Larsen and Skou [9] certainly generalizes the definition of ordinary bisimulation in a nice way, although at the cost of leaving out the categorical scenario discussed above. However, Vink and Rutten proved in [17] that the definition can be reformulated in a coalgebraic way. For that, they considered a functor $\mathcal{D}$ defining probabilistic distributions, that appears as the primitive construction in the definition of the corresponding probabilistic systems. Even though this is quite an elegant characterization, it forces us to leave the realm of (probabilistic) transition systems, moving into the more abstract one of probabilistic distributions.

We would like to directly manage probabilistic transition systems in order to compare the results about ordinary transition systems and those on probabilistic systems as much as possible. We have found that multi-transition systems, where we can have several identical transitions and the number of times they appear matters, constitute the adequate framework to establish the relation between those two kinds of transition systems. As a matter of fact, we will see that the use of multisets instead of just plain sets leads us to a natural presentation of relation lifting for that construction; besides, we can add the corresponding functor to the collection defining polynomial functors, thus obtaining an enlarged class with nice properties similar to those in the original class.

Although a general theory combining non-deterministic and probabilistic choices seems quite hard to develop, since it is difficult to combine both functors in a smooth way [16], we will present the case of *alternating*[1] probabilistic systems. In those systems, the classical definitions of ordinary and probabilistic bisimulation can be combined to obtain the natural definition of alternating probabilistic bisimulation, that perfectly fits into our framework based on categorical simulations on our multi-transition systems.

The functors defining ordinary transition systems and probabilistic systems can be obtained by applying an adequate *natural transformation* to a functor defining multiset transition systems. In both cases bisimulations are preserved in both directions when applying those transformations. This leads us to the general theory that we investigate in this paper: as is well-known, any natural transformation between two functors F and G transforms F-bisimulations into G-bisimulations; in addition, and more interesting, whenever the natural transformation relating F

[1] Although we call alternating to our systems, we do not need the strict alternation between non-deterministic and probabilistic states as appears in [4], but only that these two kind of choices do not appear mixed after the same state.

and G is an epi, we can reflect G-bisimulations and express them at the level of the functor F, though this cannot be done in general just by means of F-bisimulations. However, they can be characterized by using Hughes and Jacobs' notion of simulation [6], when we consider as the order on the functor F the equivalence induced by the epi-mono decomposition of the natural transformation relating F and G. Once categorical simulations have come into play, it is nice to find that we can extend all our results to simulations based on any order. These extensions can be considered to be the main results in the paper, since all our previous results on bisimulations could be presented as particular cases of them, using the fact that bisimulations are a particular case of categorical simulations.

Although in a different direction, namely, that of exploring the relation between non-deterministic and probabilistic choices instead of the different notions of distributed bisimulations, in this paper we continue the work initiated in FORTE 2007 [3]. The goal is the exploration of ways in which the general theory of categorical bisimulations and simulations can be applied to obtain almost for free interesting results on concrete cases that, without the support of that general theory, would need different non-trivial proofs. Therefore, our work has a mixed flavour: on the one hand we develop new abstract results that extend the general theory; on the other hand we apply these results to simple but important concrete concepts, that therefore are proved to be particular cases of the rich general theory. These are only concrete examples that we hope to extend and generalize in the near future.

2 Basic Definitions

We review in this section standard material on coalgebras and bisimulations, as can be found for example in [8,12,7]. Besides, we introduce some notations on multisets and the corresponding functor $\mathcal{M}$, as well as for the functor $\mathcal{D}$ defining discrete probabilistic distributions.

An arbitrary endofunctor $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ can be lifted to a functor in the category $\mathbf{Rel}$ of relations $\mathrm{Rel}(F) : \mathbf{Rel} \longrightarrow \mathbf{Rel}$. In set-theoretic terms, for a relation $R \subseteq X_1 \times X_2$,

$$\mathrm{Rel}(F)(R) = \{\langle u, v\rangle \in FX_1 \times FX_2 \mid \exists w \in F(R).\, F(r_1)(w) = u, F(r_2)(w) = v\}\,.$$

It is well-known that for polynomial functors F, $\mathrm{Rel}(F)$ can be equivalently defined by induction on the structure of F. Since we will be making extensive use of the powerset functor, we next present how the definition particularizes to it:

$$\begin{aligned}\mathrm{Rel}(\mathcal{P}G)(R) = \{(U, V) \mid\ & \forall u \in U.\, \exists v \in V.\, \mathrm{Rel}(G)(R)(u, v) \wedge \\ & \forall v \in V.\, \exists u \in U.\, \mathrm{Rel}(G)(R)(u, v)\}\,.\end{aligned}$$

Multisets will be represented by considering their characteristic function $\chi_M : X \longrightarrow \mathbb{N}$; similarly, discrete probabilistic distributions are represented by discrete measures $p_D : X \longrightarrow [0, 1]$, with $\sum_{x\in X} p_D(x) = 1$.

We will use along the paper several different ways to enumerate the "elements" of a multiset. We define the support of a multiset M as the set of elements that appear in it: $\{M\}_X = \{x \in X \mid \chi_M(x) > 0\}$. We are only interested in multisets having a finite support, so that in the following we will assume that every multiset is finite. Given a finite subset Y of X and an enumeration of its elements $\{y_1, \ldots y_m\}$, for each tuple of natural weights $\langle n_1, \ldots, n_m\rangle$ we will denote by $\sum_{y_i \in Y} n_i \cdot y_i$ the multiset M given by $\chi_M(y_i) = n_i$ and $\chi_M(y) = 0$ for $y \notin Y$. By abuse of notation we will sometimes consider sets as a particular case of multisets, by taking for each finite set $Y = \{y_1, \ldots y_n\}$ the canonical associated multiset $\sum_{y_i \in Y} 1 \cdot y_i$. Finally, we also enumerate the elements of a multiset by means of a generating function: given a finite set I and $x : I \longrightarrow X$, we denote by $\{x_i \mid i \in I\}$ the multiset M_I given by $\chi_{M_I}(y) = |\{i \in I \mid x_i = y\}|$. Note that in this case sets are just the multisets generated by an injective generating function.

We will denote by $\mathcal{M}(X)$ the set of multisets on X, while $\mathcal{D}(X)$ represents the set of probabilistic distributions on X. Both constructions can be naturally extended to functions, thus getting the desired functors: for $f : X \longrightarrow Y$ we define $\mathcal{M}(f) : \mathcal{M}(X) \longrightarrow \mathcal{M}(Y)$ by $\mathcal{M}(f)(\chi)(y) = \sum_{f(x)=y} \chi(x)$, and $\mathcal{D}(f) : \mathcal{D}(X) \longrightarrow \mathcal{D}(Y)$ by $\mathcal{D}(f)(p)(y) = \sum_{f(x)=y} p(x)$.

Although the multiset and the probabilistic distributions functors are not polynomial, this class can be enlarged by incorporating them since their liftings can be defined with the following equations:

$$\mathrm{Rel}(\mathcal{M}G)(R) = \{(M, N) \mid \exists f : I \longrightarrow GX, g : I \longrightarrow GY, \text{generating functions of } M \text{ and } N \text{ s.t. } \forall i \in I.\, (f(i), g(i)) \in \mathrm{Rel}(G)(R)\}\,;$$

$$\mathrm{Rel}(\mathcal{D}G)(R) = \{(d^x, d^y) \in \mathcal{D}(G(X)) \times \mathcal{D}(G(Y)) \mid \forall U \subseteq G(X).\, \forall V \subseteq G(Y).\, \Pi_1^{-1}(U) = \Pi_2^{-1}(V) \Rightarrow \textstyle\sum_{x \in U} d^x(x) = \sum_{y \in V} d^y(y)\}\,,$$

where Π_1 and Π_2 are the projections of $\mathrm{Rel}(G)(R)$ into GX and GY, respectively.

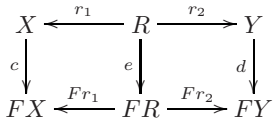
F-coalgebras are just functions $\alpha : X \longrightarrow FX$. For instance, plain labelled transition systems arise as coalgebras for the functor $\mathcal{P}(A \times X)$. We will also consider multitransition systems, which correspond to the functor $\mathcal{M}(A \times X)$, and probabilistic transition systems, corresponding to $\mathcal{M}_1([0,1] \times A \times X)$, where we only allow multisets in which the sum of its associated probabilities is 1.

Then, the lifting of the functor $\mathcal{M}_1([0,1] \times \cdot)$ is defined as a particular case of that of $\mathcal{M}$ by:

$$\begin{aligned}&\mathrm{Rel}(\mathcal{M}_1([0,1] \times \cdot)G)(R) = \\ &\quad \{(M, N) \in \mathcal{M}_1([0,1] \times GX) \times \mathcal{M}_1([0,1] \times GY) \mid \\ &\quad\quad \exists f : I \to [0,1] \times GX, g : I \to [0,1] \times GY, \text{generating functions of } M \text{ and} \\ &\quad\quad N \text{ s.t. } \forall i \in I.\, \Pi_1(f(i)) = \Pi_1(g(i)) \wedge (\Pi_2(f(i)), \Pi_2(g(i))) \in \mathrm{Rel}(G)(R)\}\,.\end{aligned}$$

A bisimulation for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ which is "closed under c and d": if $(x, y) \in R$ then $(c(x), d(y)) \in \mathrm{Rel}(F)(R)$. We shall use the term F-bisimulation sometimes to emphasize the functor we are working with.

Bisimulations can also be characterized by means of spans, using the general categorical definition by Aczel and Mendler [1]:



R is a bisimulation iff it is the carrier of some coalgebra e making the above diagram commute, where the r_i are the projections of R into X and Y.

We will also need the general concept of simulation introduced by Hughes and Jacobs [6] using orders on functors. Let $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ be a functor. An order on F is defined by means of a functorial collection of preorders $\sqsubseteq_X \subseteq FX \times FX$ that must be preserved by renaming: for every $f : X \longrightarrow Y$, if $u \sqsubseteq_X u'$ then $Ff(u) \sqsubseteq_Y Ff(u')$.

Given an order $\sqsubseteq$ on F, a $\sqsubseteq$-simulation for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ such that

$$\text{if } (x, y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)_{\sqsubseteq}(R)\,,$$

where $\mathrm{Rel}(F)_{\sqsubseteq}(R)$ is $\sqsubseteq \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq$, which can be expanded to

$$\mathrm{Rel}(F)_{\sqsubseteq}(R) = \{(u, v) \mid \exists w \in F(\mathcal{R}).\ u \sqsubseteq Fr_1(w) \wedge Fr_2(w) \sqsubseteq v\}\,.$$

One of the cases under this general notion of coalgebraic simulation is that of ordinary simulation. Also, equivalence (functorial) relations, represented by $\equiv$, are a particular class of orders on F, thus generating the corresponding class of $\equiv$-simulations. As is the case for ordinary bisimulations, $\equiv$-simulations themselves need not be equivalence relations, but once we impose to the equivalence $\equiv$ the technical condition of being stable [6] then the induced notion of $\equiv$-similarity becomes an equivalence itself.

Proposition 1. *For any stable functorial equivalence relation $\equiv_X \subseteq FX \times FX$, the induced notion of $\equiv_a$-similarity relating elements of X for a coalgebra $a : X \longrightarrow FX$ is an equivalence relation. In particular, for the plain equality relation $=_X \subseteq FX \times FX$, $=_X$-similarity coincides with plain F-bisimulation.*

3 Natural Transformations and Bisimulations

Natural transformations are the natural way to relate two functors. Given F and G, two functors on **Sets**, a natural transformation $\alpha : F \Rightarrow G$ is defined as a family of functions $\alpha_X : FX \to GX$ such that, for all $f : X \longrightarrow Y$, $Gf \circ \alpha_X = \alpha_Y \circ Ff$. We are particularly interested in the natural transformations relating $\mathcal{M}$ and $\mathcal{P}$, and those between the functors defining probabilistic transition systems and probabilistic distributions. For the sake of conciseness we will often omit the action component A when working with these functors; this does not affect the validity of the definitions nor the results.

Proposition 2. *The support of multisets,* $\{\cdot\}_X : \mathcal{M}(X) \longrightarrow \mathcal{P}(X)$*, gives rise to a natural transformation* $\{\cdot\} : \mathcal{M} \Rightarrow \mathcal{P}$*.*

Similarly, $\mathcal{D}_{M_X} : \mathcal{M}_1([0,1] \times X) \longrightarrow \mathcal{D}(X)$ *given by*

$$\mathcal{D}_M(\sum n_i \cdot (p_i, x_i))(x) = \sum_{x_i = x} n_i p_i$$

induces a natural transformation $\mathcal{D}_M : \mathcal{M}_1([0,1] \times \cdot) \Rightarrow \mathcal{D}(\cdot)$*.*

Proof. Let $f : X \longrightarrow Y$. We have $(\mathcal{P}f \circ \{\cdot\}_X)(\sum n_i \cdot x_i) = \mathcal{P}f(\{x_i\}) = \{f(x_i)\} = \{\cdot\}_Y(\sum n_i \cdot f(x_i)) = (\{\cdot\}_Y \circ \mathcal{M}f)(\sum n_i \cdot x_i)$, which proves that $\{\cdot\}$ is a natural transformation.

In the case of $\mathcal{D}_M$: $(\mathcal{D}f \circ \mathcal{D}_{M_X})(\sum n_i \cdot (p_i, x_i)) = \mathcal{D}f(\sum n_i p_i \cdot x_i)$, which is $\sum_{f(x_i)=y} n_i p_i \cdot y = \mathcal{D}_{M_Y}(\sum n_i \cdot (p_i, f(x_i))) = (\mathcal{D}_{M_Y} \circ \mathcal{M}_1 f)(\sum n_i \cdot (p_i, x_i))$; this proves that $\mathcal{D}_M$ is a natural transformation. □

Probabilistic transition systems were defined in [9] as $\mathcal{P} = (Pr, Act, Can, \mu)$, where Pr is a set of processes, Act the set of actions, $Can : Pr \longrightarrow \mathcal{P}(Act)$ indicates the initial offer of each process, and $\mu_{p,a} \in \mathcal{D}(Pr)$ for all $p \in Pr$, $a \in Can(p)$. Under this definition we cannot talk about "different probabilistic transitions" reaching the same process, that is, whenever we have a transition $p \xrightarrow{a}_\mu p'$ it "accumulates" all possible ways to go from p to p' executing a.

In our opinion this is not a purely operational way to present probabilistic systems. For instance, if we are defining the operational semantics of a process such as $p = \frac{1}{2}a + \frac{1}{2}a$, then we would intuitively have two different transitions reaching the same final state *stop*, but if we were using Larsen and Skou's original definition, we should mix them both into a single $p \xrightarrow{a}_1 stop$. Certainly, we could keep these two transitions separated under that definition if, for some reason, we decided to introduce in the set Pr two different states $stop_1$ and $stop_2$, thus obtaining $p \xrightarrow{a}_{1/2} stop_1$ and $p \xrightarrow{a}_{1/2} stop_2$. But then we observe that whether our model captures or not the existence of two different transitions depends on the way we define our set of processes Pr.

In order to get a more natural operational representation of probabilistic systems we define them[2] as $\mathcal{M}_1([0,1] \times A \times \cdot)$-coalgebras. Once we use "ordinary" transitions labelled by pairs (q, a) to represent the probabilistic transitions we have no problem to distinguish two "different" transitions $p \xrightarrow{a}_{q'} p'$, $p \xrightarrow{a}_{q''} p''$, if $p' \neq p''$. However, in such a case it would not be adequate to treat the case $p' = p''$ in a different way. This is why we use $\mathcal{M}_1$ instead of $\mathcal{P}_1$ to define our probabilistic multi-transition systems (abbreviated as pmts).

We can easily translate the classical definition of probabilistic bisimulation between probabilistic transition systems in [9], to our own pmts's as follows.

[2] Although Larsen and Skou defined their systems following the reactive aproach [4], and therefore the sum of their probabilities is 1 for each action a, we prefer to follow in this paper the generative aproach, so that the total addition of all the probabilities is 1. This is done to simplify the notation, since all the results in this paper are equally valid for the reactive model.

Definition 1. *A probabilistic bisimulation on a coalgebra $p : X \to \mathcal{M}_1([0,1] \times A \times X)$ is an equivalence relation $\equiv_p$ on X such that, whenever $x_1 \equiv_p x_2$, taking $p(x_i) = \sum t_j^i \cdot (p_j^i, a_j^i, x_j^i)$, we also have $\sum\{t_j^1 \cdot p_j^1 \mid a_j^1 = a,\, x_j^1 \in E\} = \sum\{t_j^2 \cdot p_j^2 \mid a_j^2 = a,\, x_j^2 \in E\}$, for all $a \in A$ and every equivalence class E in $X/\equiv_p$.*

In [17] it is proved that probabilistic bisimilarity defined by probabilistic bisimulations coincides with categorical $\mathcal{D}$-bisimilarity. By applying the functor $\mathcal{D}_M$ we can transform our pmts's into their presentation as Larsen and Skou's pts's. Then it is trivial to check that the corresponding notions of probabilistc bisimulation coincide, and therefore they also coincide with categorical $\mathcal{D}$-bisimilarity.

However, that is clearly not the case for plain categorical $\mathcal{M}_1([0,1] \times A \times \cdot)$-bisimulations. This is so because when we consider the functor $\mathcal{M}_1([0,1] \times A \times \cdot)$, probabilistic transitions are considered as plain transitions labelled with pairs over $[0,1] \times A$, whose first component has no special meaning. As a result, we have, for instance, no bisimulation relating x and y if we consider $X = \{x\}$, $Y = \{y\}$, $p_a : X \to \mathcal{M}_1([0,1] \times A \times X)$ with $p_a(x) = 1 \cdot (1, a, x)$ and $p_b : Y \to \mathcal{M}_1([0,1] \times A \times Y)$ with $p_b(y) = 2 \cdot (\frac{1}{2}, a, y)$.

All these facts prove that our probabilistic multi-transition systems are too concrete a representation of probabilistic distributions, which is formally captured by the fact that the components of the natural transformation $\mathcal{D}_M$ are not injective. As a consequence, by using them we do not have a pure coalgebraic characterization of probabilistic bisimulations. By contrast, the original definition of pts's stands apart from the operational way, mixing different transitions into a single distribution. Besides it has to consider the quotient set $X/\equiv_p$ when defining probabilistic bisimulations. Our goal will be to obtain a characterization of the notion of probabilistic bisimilarity in terms of our pmts's, and this will be done using the notion of categorical simulation, as we will see in Section 4. Next, we present a collection of general interesting results. First we will see that bisimulations are preserved by natural transformations.

Theorem 1 ([12]). *If $R \subseteq X \times Y$ is a bisimulation relating $a : X \longrightarrow FX$ and $b : Y \longrightarrow FY$, then R is also a bisimulation relating $a' : X \longrightarrow GX$, given by $a' = \alpha_X \circ a$, and $b' : Y \longrightarrow GY$, given by $b' = \alpha_Y \circ b$.*

Corollary 1. *For a and $a' = \alpha_X \circ a$, bisimulation equivalence in a is included in bisimulation equivalence in a', that is, $x_1 \equiv_a x_2$ implies $x_1 \equiv_{a'} x_2$.*

A general converse result cannot be expected because in general there is no canonical way to transform G into F. Since the main objective in this paper is to relate $\mathcal{M}$-bisimulations with $\mathcal{P}$ and $\mathcal{D}$-bisimulations, we searched for particular properties of the natural transformations relating these functors which could help us to get the desired general results covering in particular these two cases. This is how we have obtained the concept of quotient functors that we develop in the following.

Definition 2. *Let F be an endofunctor on* **Sets** *and $\equiv$ a functorial equivalence relation $\equiv_X \subseteq FX \times FX$. We define the* quotient functor $F/\equiv$ *by $(F/\equiv)(X) = FX/\equiv_X$, and for any $f : X \longrightarrow Y$, $u \in FX$, and $\overline{u}$ its equivalence class, $(F/\equiv)(f)(\overline{u}) = \overline{F(f)(u)}$, that is well defined since $\equiv$ is functorial.*

Definition 3. *1. We say that a functor G is the quotient of F under a functorial equivalence relation $\equiv$ whenever $F/\equiv$ and G are isomorphic, which means that there is a pair of natural transformations $\alpha : F/\equiv \Rightarrow G$ and $\beta : G \Rightarrow F/\equiv$ such that $\beta \circ \alpha = Id_{F/\equiv}$ and $\alpha \circ \beta = Id_G$.*

2. Given a natural transformation $\alpha : F \Rightarrow G$, we write $\equiv^\alpha$ for the family of equivalence relations $\equiv^\alpha_X \subseteq FX \times FX$ defined by the kernel of α: $u_1 \equiv^\alpha_X u_2 \iff \alpha_X(u_1) = \alpha_X(u_2)$.

Proposition 3. *For every natural transformation $\alpha : F \Rightarrow G$, $\equiv^\alpha$ is functorial.*

Proof. We need to show that, for any $f : X \longrightarrow Y$, whenever $u_1 \equiv^\alpha_X u_2$, that is, $\alpha_X(u_1) = \alpha_X(u_2)$, we also have $Ff(u_1) \equiv^\alpha_Y Ff(u_2)$, that is $\alpha_Y(F(f)(u_1)) = \alpha_Y(F(f)(u_2))$; this follows because $\alpha_Y \circ F(f) = G(f) \circ \alpha_X$. □

If every component α_X of a natural transformation is surjective, α is said to be epi.

Proposition 4. *Whenever α is epi, G is the quotient of F under $\equiv^\alpha$, just considering the inverse natural transformation $\alpha^{-1} : G \Rightarrow F/\equiv$ given by $\alpha^{-1}_X : G(X) \longrightarrow (F/\equiv^\alpha)(X)$ with $\alpha^{-1}_X(v) = \overline{u}$ where $\alpha_X(u) = v$.*

Corollary 2. *$\mathcal{P}$ is the quotient of $\mathcal{M}$ under the kernel of the natural transformation $\{\cdot\} : \mathcal{M} \Rightarrow \mathcal{P}$.*

Corollary 3. *$\mathcal{D}$ is the quotient of $\mathcal{M}_1([0,1] \times \cdot)$ under the kernel of the natural transformation $\mathcal{D}_{\mathcal{M}} : \mathcal{M}_1([0,1] \times \cdot) \Rightarrow \mathcal{D}$.*

4 $\equiv^\alpha$-simulations Through Quotients of Bisimulations

Let us start by studying the relationships between coalgebras corresponding to functors related by an epi natural transformation.

Definition 4. *Let $\alpha : F \Rightarrow G$ be a natural transformation and $a : X \longrightarrow FX$ an F-coalgebra. We define the α-image of a as the coalgebra $a_\alpha : X \longrightarrow GX$ given by $a_\alpha = \alpha_X \circ a$.*

Definition 5. *Given a natural transformation $\alpha : F \Rightarrow G$ and a G-coalgebra $b : X \longrightarrow GX$, we say that $a : X \longrightarrow FX$ is a concrete F-representation of b iff $b = \alpha_X \circ a$.*

The following result follows immediately from the previous definitions.

Proposition 5. *If α is epi then every G-coalgebra has an F-representation.*

Next we relate G-bisimulations with $\equiv^{\alpha}$-simulations:

Theorem 2. *Let $\alpha : F \Rightarrow G$ be an epi natural transformation and $b_1 : X_1 \longrightarrow GX_1$, $b_2 : X_2 \longrightarrow GX_2$ two G-coalgebras, with concrete F-representations $a_1 : X_1 \longrightarrow FX_1$ and $a_2 : X_2 \longrightarrow FX_2$. Then, the G-bisimulations relating b_1 and b_2 are precisely the $\equiv^{\alpha}$-simulations relating a_1 and a_2.*

Proof. Let us show[3] that, for every relation $R \subseteq X_1 \times X_2$,

$$\mathrm{Rel}(F)_{\equiv^{\alpha}}(R) = \{(u,v) \in FX_1 \times FX_2 \mid (\alpha_{X_1}(u), \alpha_{X_2}(v)) \in \mathrm{Rel}(G)(R)\}\,.$$

We have, unfolding the definition of $\mathrm{Rel}(F)_{\equiv^{\alpha}}(R)$ and using the fact that α is a natural transformation:

$$\begin{aligned}
\mathrm{Rel}(F)_{\equiv^{\alpha}}(R) &= \{(u,v) \in FX_1 \times FX_2 \mid \exists w \in FR.\, u \equiv^{\alpha} Fr_1(w) \wedge Fr_2(w) \equiv^{\alpha} v\} \\
&= \{(u,v) \in FX_1 \times FX_2 \mid \exists w \in FR.\, \alpha_{X_1}(u) = \alpha_{X_1}(Fr_1(w)) \wedge \\
&\qquad\qquad \alpha_{X_2}(v) = \alpha_{X_2}(Fr_2(w))\} \\
&= \{(u,v) \in FX_1 \times FX_2 \mid \exists w \in FR.\, \alpha_{X_1}(u) = Gr_1(\alpha_R(w)) \wedge \\
&\qquad\qquad \alpha_{X_2}(v) = Gr_2(\alpha_R(w))\}\,.
\end{aligned}$$

On the other hand,

$$\mathrm{Rel}(G)(R) = \{(x,y) \in GX_1 \times GX_2 \mid \exists z \in GR.\, Gr_1(z) = x \wedge Gr_2(z) = y\}\,.$$

Now, if $(u,v) \in \mathrm{Rel}(F)_{\equiv^{\alpha}}(R)$, by taking $\alpha_R(w)$ as the value of $z \in GR$ we have that $(\alpha_{X_1}(u), \alpha_{X_2}(v)) \in \mathrm{Rel}(G)(R)$. Conversely, if $(\alpha_{X_1}(u), \alpha_{X_2}(v)) \in \mathrm{Rel}(G)(R)$ is witnessed by z, let $w \in FR$ be such that $\alpha_R(w) = z$, which must exists because α is epi; it follows that $(u,v) \in \mathrm{Rel}(F)_{\equiv^{\alpha}}(R)$.

Then, $(b_1(x), b_2(y)) \in \mathrm{Rel}(G)(R)$ if and only if $(a_1(x), a_2(x)) \in \mathrm{Rel}(F)_{\equiv^{\alpha}}(R)$, from where it follows that R is a G-bisimulation if and only if it is a $\equiv^{\alpha}$-simulation. □

Corollary 4. *(i) Bisimulations between labelled transition systems are $\equiv^{\{\cdot\}}$-simulations between multi-transition systems. (ii) Bisimulations between probabilistic systems are just $\equiv^{\mathcal{D}_M}$-simulations between (an appropriate class of) multi-transition systems.*

Example 1. Let us illustrate this result by means of some simple examples using the natural transformation $\{\cdot\} : \mathcal{M} \to \mathcal{P}$.

1. If we consider the ordinary transition systems $s_X : \{x, x'\} \longrightarrow \mathcal{P}(\{x, x'\})$, with $s_X(x) = \{x'\}$, $s_X(x') = \emptyset$, and $s_Y : \{y, y'_1, y'_2\} \longrightarrow \mathcal{P}(\{y, y'_1, y'_2\})$ with $s_Y(y) = \{y'_1, y'_2\}$, $s_Y(y'_1) = \emptyset$, and $s_Y(y'_2) = \emptyset$, we have a simple $\mathcal{P}$-bisimulation relating the initial states x and y, given by $R = \{(x,y), (x', y'_1), (x', y'_2)\}$.

[3] It is not difficult to present this proof as a commutative diagram. Then one has to check that all the "small squares" in the diagram are indeed commutative, in order to be able to conclude commutativity of the full diagram. This is what we have carefully done in our proof above.

Denoting by s^1_X and s^1_Y the canonical $\mathcal{M}$-representations of s_X and s_Y, obtained by the embedding of sets into multisets, it is obvious that there is no $\mathcal{M}$-bisimulation relating x and y. But if we consider $s^2_X(x) = \{2 \cdot x'\}$, $s^2_X(x') = \emptyset$, we have now an $\mathcal{M}$-bisimulation between the multi-transition systems s^2_X and s^1_Y relating x and y. And, by Theorem 2, we have that s^1_X is also $\equiv^{\{\cdot\}}$-simulated by s^1_Y, since $\{s^1_X\}_{\mathcal{M}} = \{s^2_X\}_{\mathcal{M}} = s_X$ and s_X and s_Y are $\mathcal{P}$-bisimilar. Obviously, the same happens for any $\{\cdot\}$-representation of s_X, s^k_X with $s^k_X = \{k \cdot x'\}$ and $s^k_X(x') = \emptyset$.

2. In the example above we got the $\equiv^{\{\cdot\}}$-simulation by proving that there are $\mathcal{M}$-representations of the considered coalgebras for which the given relation is also an $\mathcal{M}$-bisimulation. However, this is not necessary as the following shows. Let us consider $t_X : \{x\} \longrightarrow \mathcal{P}(\{x\})$ with $t_X(x) = \{x\}$ and $Y = \{\beta \mid \beta \in \mathbb{N}^*, \beta_i \leq i\}$ with $t_Y(\beta) = \{\beta \circ \langle j \rangle \mid \beta \circ \langle j \rangle \in Y\}$. It is clear that $R = \{(x, \beta) \mid \beta \in Y\}$ is the (only) $\mathcal{P}$-bisimulation relating x and ϵ, the initial states of t_X and t_Y. However, in this case there exists no $\mathcal{M}$-bisimulation relating two $\mathcal{M}$-representations of t_X and t_Y, because $|t_Y(\beta)| = |\beta| + 1$ and therefore we would need a representation t^k_X with $t^k_X(x) = \{k \cdot x\}$ such that $k \geq l$ for all $l \in \mathbb{N}$, which is not possible because the definition of multiset does not allow the infinite repetition of any of its members. Instead, Theorem 2 shows that any two $\mathcal{M}$-representations of t_X and t_Y are $\equiv^{\{\cdot\}}$-similar.

The reason why we had an $\mathcal{M}$-bisimulation relating the appropriate $\mathcal{M}$-representations of the compared $\mathcal{P}$-coalgebras in our first example was because we were under the hypothesis of the following proposition.

Proposition 6. *Let $\alpha : F \Rightarrow G$ be an epi natural transformation. Whenever a G-bisimulation R relating $b_1 : X \longrightarrow GX$ and $b_2 : Y \longrightarrow GY$ is* near injective*, which means that $|\{b_2(y) \mid (x, y) \in R\}| \leq 1$ for all $x \in X$ and $|\{b_1(x) \mid (x, y) \in R\}| \leq 1$ for all $y \in Y$, there exist some F-representations of b_1 and b_2, $a_1 : X \longrightarrow FX$ and $a_2 : Y \longrightarrow FY$, respectively, such that R is also a bisimulation relating a_1 and a_2.*

Proof. By Theorem 2, R is also a $\equiv^\alpha$-simulation for any pair of F-representations of b_1 and b_2; let a_1, a_2 be any such pair. Then, for all $(x, y) \in R$ we have $(a_1(x), a_2(y)) \in (\equiv^\alpha \circ \mathrm{Rel}(F) \circ \equiv^\alpha)(R)$, and hence there exist $a'_1(x, y) \in FX$, $a'_2(x, y) \in FY$ such that

$$a_1(x) \equiv^\alpha a'_1(x, y),\ a'_2(x, y) \equiv^\alpha a_2(y) \text{ and } (a'_1(x, y), a'_2(x, y)) \in \mathrm{Rel}(F)(R)\,.$$

We now define an equivalence relation $\equiv$ on R by considering the transitive closure of:

- $(x, y_1) \equiv (x, y_2)$ for all $(x, y_1), (x, y_2) \in R$.
- $(x_1, y) \equiv (x_2, y)$ for all $(x_1, y), (x_2, y) \in R$.

Since R is near injective, it follows that if $(x_1, y_1) \equiv (x_2, y_2)$ then $b_1(x_1) = b_1(x_2)$ and $b_2(y_1) = b_2(y_2)$, and thus $a'_1(x_1, y_1) \equiv^\alpha a'_1(x_2, y_2)$ and $a'_2(x_1, y_1) \equiv^\alpha a'_2(x_2, y_2)$.

We consider $R/\equiv$ and for each equivalence class of the quotient set we choose a canonical representative $\overline{(x,y)}$. Obviously we have that $(x,y_1),(x,y_2) \in R$ implies $\overline{(x,y_1)} = \overline{(x,y_2)}$ and that $(x_1,y),(x_2,y) \in R$ implies $\overline{(x_1,y)} = \overline{(x_2,y)}$.

Let us now define two coalgebras $a'_1 : X \longrightarrow FX$ and $a'_2 : Y \longrightarrow FY$ as follows:

- If there exists some y such that $(x,y) \in R$ we take $a'_1(x) = a'_1\overline{(x,y)}$ for any such y; otherwise, we define $a'_1(x)$ as $a_1(x)$.
- If there exists some x such that $(x,y) \in R$ we take $a'_2(y) = a'_2\overline{(x,y)}$ for any such x; otherwise, $a'_2(y)$ is $a_2(y)$.

With the above definitions,

$$a'_1(x) = a'_1\overline{(x,y)} \equiv^{\alpha} a'_1(x,y) \equiv^{\alpha} a_1(x)\,,$$

and similarly $a'_2(y) \equiv^{\alpha} a_2(y)$, so that a'_1, a'_2 are F-representations of b_1 and b_2. Besides,

$$\text{if } (x,y) \in R \text{ then } (a'_1(x), a'_2(y)) \in \mathrm{Rel}(F)(R)$$

and R is an F-bisimulation relating them. □

Let us conclude this illustration of our main theorem by explaining why we needed an infinite coalgebra to get a counterexample of the result between bisimulations relating G-coalgebras and those relating their F-representations. As a matter of fact, in the case of the multiset and the powerset functors we could prove the result in Proposition 6 not only for near injective bisimulations but for any relation where no element is related with infinitely many others. However, we will not prove this fact here since it does not seem to generalize to arbitrary natural transformations relating two functors.

Example 2. Next we present an example for the natural transformation $\mathcal{D}_M : \mathcal{M}_1([0,1] \times X) \Rightarrow \mathcal{D}(X)$. If we consider the two probabilistic transition systems s_X and s_Y given by their multisets of probabilistic transitions: $s_X = \{(\frac{1}{2},x,x'_1),(\frac{1}{2},x,x'_2)\}$, $s_Y = \{(\frac{1}{3},y,y'_1),(\frac{1}{3},y,y'_2),(\frac{1}{3},y,y'_3)\}$, where each triple (p,x,x') represents the probabilistic transition $x \xrightarrow{p} x'$, we have the following $\mathcal{D}$-bisimulation relating the initial states x and y: $R = \{(x,y)\} \cup \{(x'_i,y'_j) \mid i = 1,2, j = 1,2,3\}$. It is easy to see that for the two $\mathcal{M}_1$-representations $s^3_X = \{3\cdot(\frac{1}{6},x,x'_1), 3\cdot(\frac{1}{6},x,x'_2)\}$ and $s^2_Y = \{2\cdot(\frac{1}{6},y,y'_1), 2\cdot(\frac{1}{6},y,y'_2), 2\cdot(\frac{1}{6},y,y'_3)\}$, R is also an $\mathcal{M}_1$-bisimulation between them, using the facts that $(x'_1,y'_1) \in R$, $(x'_2,y'_2) \in R$ and $(x'_1,y'_3) \in R$, $(x'_2,y'_3) \in R$. From this result we immediately conclude that any two $\mathcal{M}_1$-representations of s_X and s_Y are $\equiv^{\mathcal{D}\mathcal{M}}$-similar.

5 Natural Transformations and Simulations

In this section we will see that all our results about bisimulations in the previous sections can be extended to categorical simulations defined by means of an order on the corresponding functors. Therefore, our first result concerns the preservation of functorial orders by means of natural transformations.

Definition 6. *Given a natural transformation $\alpha : F \Rightarrow G$ and $\sqsubseteq_G$ an order on G, we define the induced order $\sqsubseteq_G^{\alpha-}$ on F by: $x \sqsubseteq_G^{\alpha-} x' \iff \alpha_X(x) \sqsubseteq_G \alpha_X(x')$.*

It is immediate that $\sqsubseteq_G^{\alpha-}$ is indeed an order on F; given $f : X \longrightarrow Y$ and $x, x' \in X$:

$$\begin{aligned} x \sqsubseteq_G^{\alpha-} x' &\iff \alpha_X(x) \sqsubseteq_G \alpha_X(x') \\ &\implies Gf(\alpha_X(x)) \sqsubseteq_G Gf(\alpha_X(x')) \\ &\iff \alpha_Y(Ff(x)) \sqsubseteq_G \alpha_Y(Ff(x')) \\ &\iff Ff(x) \sqsubseteq_G^{\alpha-} Ff(x'), \end{aligned}$$

where the implication follows because $\sqsubseteq_G$ is functorial.

Example 3. Taking $\{\cdot\} : \mathcal{M} \Rightarrow \mathcal{P}$ and $\sqsubseteq_{\mathcal{P}} = \subseteq$, then the induced order $\sqsubseteq_{\mathcal{P}}^{\{\cdot\}-}$ on $\mathcal{M}$ is defined as $u \sqsubseteq_{\mathcal{P}}^{\{\cdot\}-} v$ iff $\{u\} \subseteq \{v\}$: that is, it coincides with multiset inclusion.

Another example corresponds to the equality relation on G.

Proposition 7. *The induced order $=_G^{\alpha-}$ on F is just the relation $\equiv^{\alpha}$.*

Proof. The definition of $\equiv^{\alpha}$ is just the particular case of our definition of $\sqsubseteq_G^{\alpha-}$ for the equality relation on G as an order on it. □

Orders on F can be also translated to G through a natural transformation $\alpha : F \Rightarrow G$.

Definition 7. *Given a natural transformation $\alpha : F \Rightarrow G$ and $\sqsubseteq_F$ an order on F, we define the projected order $\sqsubseteq_F^{\alpha}$ on G as the transitive closure of the relation $x \sqsubseteq_F^{\alpha} x'$, which holds if:*

there exist x_1, x_1' such that $x = \alpha_X(x_1)$, $x' = \alpha_X(x_1')$ and $x_1 \sqsubseteq_F x_1'$, or $x = x'$.

We need to add the last condition in the definition above in order to cover the case in which α is not an epi. Obviously, we can remove it whenever α is indeed an epi, and in the following we will see that we only need that condition in order to guarantee reflexivity of $\sqsubseteq_F^{\alpha}$ in the whole of GX, because all of our results concerning this order will be based on its restriction to the images of the components of the natural transformation α_X.

Again, it is easy to prove that $\sqsubseteq_F^{\alpha}$ is indeed an order on G. By definition, it is reflexive and transitive. It is also functorial: given $f : X \longrightarrow Y$ and $x \sqsubseteq_F^{\alpha} x'$, with $x = \alpha_X(x_1)$ and $x' = \alpha(x_1')$ such that $x_1 \sqsubseteq_F x_1'$, we need to show $Gf(x) \sqsubseteq_F^{\alpha} Gf(x')$. Since $Gf(x) = Gf(\alpha(x_1)) = \alpha(Ff(x_1))$, $Gf(x') = Gf(\alpha(x_1')) = \alpha(Ff(x_1'))$, and $Ff(x_1) \sqsubseteq_F Ff(x_1')$, the result follows by the definition of $\sqsubseteq_F^{\alpha}$.

Theorem 3 (Simulations are preserved by natural transformations). *If $R \subseteq X \times Y$ is a $\sqsubseteq_F$-simulation relating $a : X \longrightarrow FX$ and $b : Y \longrightarrow FY$, and $\alpha : F \Rightarrow G$ is a natural transformation, then R is also a $\sqsubseteq_F^{\alpha}$-simulation relating $a' = \alpha_X \circ a$ and $b' = \alpha_Y \circ b$.*

Proof. Let $(x, y) \in R$: we need to show that $(a'(x), b'(y)) \in \mathrm{Rel}(G)_{\sqsubseteq_F^\alpha}(R)$. Since R is a $\sqsubseteq_F$-simulation, $(a(x), b(x)) \in \mathrm{Rel}(F)_{\sqsubseteq_F}(R)$. This means that there exists $w \in FR$ such that $a(x) \sqsubseteq_F Fr_1(w)$ and $Fr_2(w) \sqsubseteq_F b(x)$, and hence that there exists $z = \alpha_R(w) \in GR$ such that $a'(x) \sqsubseteq_F^\alpha \alpha_X(Fr_1(w)) = Gr_1(z)$ and $Gr_2(z) = \alpha_Y(Fr_2(w)) \sqsubseteq_F^\alpha b'(x)$; therefore, $(a'(x), b'(x)) \in \mathrm{Rel}(G)_{\sqsubseteq_F^\alpha}(R)$. □

As said before, bisimulations are just the particular case of simulations corresponding to the equality relation. Obviously we have that $=_F^\alpha$ is $=_G$ and therefore Theorem 1 about the preservation of bisimulations by natural transformations is a particular case of our new preservation theorem covering arbitrary $\sqsubseteq_F$-simulations.

Analogously, we now generalized Theorem 2 to arbitrary $\sqsubseteq_G$-simulations.

Theorem 4. *Let $\alpha : F \Rightarrow G$ be an epi natural transformation, $\sqsubseteq_G$ an order on G and $b_1 : X_1 \longrightarrow GX_1$, $b_2 : X_2 \longrightarrow GX_2$ two coalgebras, with $a_1 : X_1 \longrightarrow FX_1$, $a_2 : X_2 \longrightarrow FX_2$ arbitrary concrete F-representations. Then, the $\sqsubseteq_G$-simulations relating b_1 and b_2 are precisely the $\sqsubseteq_G^{\alpha^-}$-simulations relating a_1 and a_2.*

Proof. Just like Theorem 2, the result follows from showing that, for every relation $R \subseteq X_1 \times X_2$,

$$\mathrm{Rel}(F)_{\sqsubseteq_G^{\alpha^-}}(R) = \{(u, v) \in FX_1 \times FX_2 \mid (\alpha_{X_1}(u), \alpha_{X_2}(v)) \in \mathrm{Rel}(G)_{\sqsubseteq_G^\alpha}(R)\}\,.$$

Unfolding the definition of $\mathrm{Rel}(F)_{\sqsubseteq_G^{\alpha^-}}(R)$ and using the fact that α is a natural transformation:

$$\begin{aligned}
\mathrm{Rel}(F)_{\sqsubseteq_G^{\alpha^-}}(R) &= \{(u, v) \in FX_1 \times FX_2 \mid \exists w \in FR.\, u \sqsubseteq_G^{\alpha^-} Fr_1(w) \wedge \\
&\qquad\qquad Fr_2(w) \sqsubseteq_G^{\alpha^-} v\} \\
&= \{(u, v) \in FX_1 \times FX_2 \mid \exists w \in FR.\, \alpha_{X_1}(u) \sqsubseteq_G \alpha_{X_1}(Fr_1(w)) \wedge \\
&\qquad\qquad \alpha_{X_2}(Fr_2(w)) \sqsubseteq_G \alpha_{X_2}(v)\} \\
&= \{(u, v) \in FX_1 \times FX_2 \mid \exists w \in FR.\, \alpha_{X_1}(u) \sqsubseteq_G Gr_1(\alpha_R(w)) \wedge \\
&\qquad\qquad Gr_2(\alpha_R(w)) \sqsubseteq_G \alpha_{X_2}(v)\}\,.
\end{aligned}$$

On the other hand,

$$\mathrm{Rel}(G)_{\sqsubseteq_G}(R) = \{(x, y) \in GX_1 \times GX_2 \mid \exists z \in GR.\, x \sqsubseteq_G Gr_1(z) \wedge Gr_2(z) \sqsubseteq_G y\}\,.$$

Now, if $(u, v) \in \mathrm{Rel}(F)_{\sqsubseteq_G^{\alpha^-}}(R)$, by taking $\alpha_R(w)$ as the value of $z \in GR$ we have that $(\alpha_{X_1}(u), \alpha_{X_2}(v)) \in \mathrm{Rel}(G)_{\sqsubseteq_G}(R)$. Conversely, if $(\alpha_{X_1}(u), \alpha_{X_2}(v)) \in \mathrm{Rel}(G)_{\sqsubseteq_G}(R)$ is witnessed by z, let $w \in FR$ be such that $\alpha_R(w) = z$, which must exist because α is epi; it follows that $(u, v) \in \mathrm{Rel}(F)_{\sqsubseteq_G^{\alpha^-}}(R)$. □

6 Combining Non-determinism and Probabilistic Choices

Probabilistic choice appears as a quantitative counterpart of non-deterministic choice. However, it has been also argued that the motivations supporting the use

of these two constructions are different, so that it is also interesting to be able to manage both together. The literature on the subject is full of proposals in this direction [13,10,14], but it has been proved in [16] that there is no distributive law of the probabilistic monad V over the powerset monad P. As a consequence, if we want to combine the two categorical theories to obtain a common framework, we have to sacrifice some of the properties of one of those monads. Varacca and Winskel have followed this idea by relaxing the definition of the monad V, removing the axiom $A \oplus_p A = A$, so that they are aware of the probabilistic choices taken along a computation even if they are superfluous.

We have not yet studied that general case, whose solution in [16] is technically correct, but could be considered intuitively not too satisfactory since one would like to maintain the idempotent law $A \oplus_p A = A$, even if this means that only some practical cases can be considered.

As a first step in this direction we will present here the simple case of alternating probabilistic systems, which in our multi-transition system framework can be defined as follows:

Definition 8. *Alternating multi-transition systems are defined as* $(\mathcal{M}(A \times \cdot) \cup \mathcal{M}_1([0,1] \times A \times \cdot))$*-coalgebras: any state of a system represents either a non-deterministic choice or a probabilistic choice; however, probabilistic and non-deterministic choices cannot be mixed together.*

By combining the two natural transformations $\{\cdot\}$ and $\mathcal{D}_M$ we obtain the natural transformation $\mathcal{D}^a_M$, that captures the behaviour of alternating transition systems.

Definition 9. *We use the term* alternating probabilistic systems *to refer to the* $(\mathcal{P}(A \times \cdot) \cup \mathcal{D}(A \times \cdot))$*-coalgebras. By combining the classical definition of bisimulation and that of probabilistic bisimulations we obtain the natural definition of probabilistic bisimulation for alternating probabilistic systems.*

We define $\mathcal{D}^a_{M_X} : \mathcal{M}(A \times \cdot) \cup \mathcal{M}_1([0,1] \times A \times \cdot) \Rightarrow \mathcal{P}(A \times \cdot) \cup \mathcal{D}(A \times \cdot)$ *as* $\mathcal{D}^a_{M_X}(M) = \{\cdot\}(M)$, $\mathcal{D}^a_{M_X}(M_1) = \mathcal{D}_M(M_1)$, *where* $M \in \mathcal{M}(A \times X)$, $M_1 \in \mathcal{M}_1([0,1] \times A \times X)$.

Then we can consider the induced functorial equivalence $\equiv^{\mathcal{D}^a_M}$ which roughly corresponds to the application of $\equiv^{\{\cdot\}}$ in the non-deterministic states, and the application of $\equiv^{\mathcal{D}_M}$ in the probabilistic states. As a consequence of Theorem 2 we obtain the following corollary.

Corollary 5. *Bisimulations between alternating probabilistic systems are just* $\equiv^{\mathcal{D}^a_M}$*-simulations between alternating multi-transition systems.*

Example 4. Let $X = \{x, x'_1, x'_2, x'_3, x'_4\}$, $Y = \{y, y'_1, y'_2, y'_3, y'_4\}$ and let us define (disregarding actions) the alternating multi-transition systems $a_X : X \longrightarrow \mathcal{M}(X) \cup \mathcal{M}_1([0,1] \times X)$ and $a_Y : Y \longrightarrow \mathcal{M}(Y) \cup \mathcal{M}_1([0,1] \times Y)$ as $a_X(x) = \{1 \cdot (\frac{1}{2}, x'_1), 1 \cdot (\frac{1}{2}, x'_2)\}$, $a_X(x'_1) = \{1 \cdot x'_3\}$, $a_X(x'_2) = \{1 \cdot x'_4\}$, $a_X(x'_3) = a_X(x'_4) = \emptyset$, $a_Y(y) = \{1 \cdot (\frac{1}{3}, y'_1), 1 \cdot (\frac{1}{3}, y'_2), 1 \cdot (\frac{1}{3}, y'_3)\}$, $a_Y(y'_1) = a_Y(y'_2) = a_Y(y'_3) = \{1 \cdot y'_4\}$,

$a_Y(y'_4) = \emptyset$. a_X and a_Y induce the canonical alternating probabilistic systems $b_X : X \longrightarrow \mathcal{P}(X) \cup \mathcal{D}(X)$ and $b_Y : Y \longrightarrow \mathcal{P}(Y) \cup \mathcal{D}(Y)$ (for example, $b_X(x) = \frac{1}{2}x'_1 + \frac{1}{2}x'_2$ and $b_Y(y'_3) = \{y'_4\}$).

Now, if we want to know if there is a bisimulation between b_X and b_Y we can use the fact that $R = \{(x, y)\} \cup \{(x'_i, y'_j) \mid i = 1, 2, j = 1, 2, 3\} \cup \{(x'_i, y'_4) \mid i = 3, 4\}$ is a $\equiv^{\mathcal{D}^a_M}$-bisimulation between a_X and a_Y (using a similar argument to that in Example 2), and apply Corollary 5 to conclude that there is a $(\mathcal{P} \cup \mathcal{D})$-bisimulation between b_X and b_Y.

7 Conclusion

In this paper we have shown that multitransition systems are a common framework wherein bisimulation of ordinary and probabilistic transition systems almost collapse into the same concept of multiset (bi)simulation. Indeed, the definition of bisimulation for the multiset functor is extremely simple, which supports the idea that multisets are the natural framework in which to justify the use of bisimulation as the canonical notion of equivalence between (states of) systems.

These results have been obtained by exploiting the fact that natural transformations between two functors relate in a nice way bisimulations over their corresponding coalgebras. We have illustrated these general results by means of the natural transformations that connect the powerset and the probabilistic distributions functors with the multiset functor.

The categorical notion of simulation proposed by Hughes and Jacobs has played a very important role in our work; this fact, in our opinion, is far from being casual. In particular, categorical simulations based on equivalence relations always define equivalence relations weaker than bisimulation equivalence. Besides, as illustrated by their use in this paper, they can be used to relate the bisimulation equivalence corresponding to functors connected by a natural transformation.

Related to our work is [2], where probabilistic bisimulations are studied in connection with natural transformations and other categorical notions. Even though some connections can be found, there are very important differences; in particular they do not consider categorical simulations nor use the multiset functor as a general framework in which to study both ordinary and probabilistic bisimulations. We can also mention [15], where the functor $\mathcal{D}$ is replaced with a functor of indexed valuations so that it can be combined with the powerset functor.

A direction for further study that we intend to explore concerns other classes of bisimulations, like the forward-backward ones estudied in [5]. Besides we will study more general combinations of non-deterministic and probabilistic choices, comparing in detail our approach with the use of indexed valuations in [15,16] to combine the monads defining the corresponding functors.

We are confident we will be able to study them in a common setting by generalizing and adapting all the appropriate notions on categorical simulations.

References

1. Aczel, P., Mendler, N.P.: A final coalgebra theorem. In: Pitt, D.H., Rydeheard, D.E., Dybjer, P., Pitts, A.M., Poigné, A. (eds.) Category Theory and Computer Science. LNCS, vol. 389, pp. 357–365. Springer, Heidelberg (1989)
2. Bartels, F., Sokolova, A., de Vink, E.P.: A hierarchy of probabilistic system types. Theoretical Computer Science 327(1-2), 3–22 (2004)
3. de Frutos-Escrig, D., Rosa-Velardo, F., Gregorio-Rodríguez, C.: New Bisimulation Semantics for Distributed Systems. In: Derrick, J., Vain, J. (eds.) FORTE 2007. LNCS, vol. 4574, pp. 143–159. Springer, Heidelberg (2007)
4. van Glabbeek, R.J., Smolka, S.A., Steffen, B.: Reactive, Generative and Stratified Models of Probabilistic Processes. Information and Computation 121(1), 59–80 (1995)
5. Hasuo, I.: Generic forward and backward simulations. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 406–420. Springer, Heidelberg (2006)
6. Hughes, J., Jacobs, B.: Simulations in coalgebra. Theoretical Computer Science 327(1-2), 71–108 (2004)
7. Jacobs, B.: Introduction to coalgebra. towards mathematics of states and observations. Book in preparation, Available at: `http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`
8. Jacobs, B., Rutten, J.: A tutorial on (co)algebras and (co)induction. Bulletin of the European Association for Theoretical Computer Science 62, 222–259 (1997)
9. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94(1), 1–28 (1991)
10. Mislove, M.W.: Nondeterminism and Probabilistic Choice: Obeying the Laws. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 350–364. Springer, Heidelberg (2000)
11. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
12. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theoretical Computer Science 249(1), 3–80 (2000)
13. Segala, R., Lynch, N.A.: Probabilistic Simulations for Probabilistic Processes. Nordic Journal on Computing 2(2), 250–273 (1995)
14. Tix, R., Keimel, K., Plotkin, G.: Semantic Domains for Combining Probability and Non-Determinism. ENTCS, vol. 129, pp. 1–104. Elsevier, Amsterdam (2005)
15. Varacca, D.: The powerdomain of indexed valuations. In: LICS 2002: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science, pp. 299–310. IEEE Computer Society, Los Alamitos (2002)
16. Varacca, D., Winskel, G.: Distributing Probabililty over Nondeterminism. Mathematical Structures in Computer Science 16(1), 87–113 (2006)
17. de Vink, E.P., Rutten, J.J.M.M.: Bisimulation for probabilistic transition systems: A coalgebraic approach. Theoretical Computer Science 221(1-2), 271–293 (1999)

Relating modal refinements, covariant-contravariant simulations and partial bisimulations*

Luca Aceto[1], Ignacio Fábregas[2], David de Frutos Escrig[2], Anna Ingólfsdóttir[1], and Miguel Palomino[2]

[1] ICE-TCS, School of Computer Science, Reykjavik University, Iceland
[2] Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Spain

Abstract. This paper studies the relationships between three notions of behavioural preorder that have been proposed in the literature: refinement over modal transition systems, and the covariant-contravariant simulation and the partial bisimulation preorders over labelled transition systems. It is shown that there are mutual translations between modal transition systems and labelled transition systems that preserve, and reflect, refinement and the covariant-contravariant simulation preorder. The translations are also shown to preserve the modal properties that can be expressed in the logics that characterize those preorders. A translation from labelled transition systems modulo the partial bisimulation preorder into the same model modulo the covariant-contravariant simulation preorder is also offered, together with some evidence that the former model is less expressive than the latter. In order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, their connections are also phrased and studied in the context of institutions.

1 Introduction

Modal transition systems (MTSs) have been proposed in, e.g., [11,12] as a model of reactive computation based on states and transitions that naturally supports a notion of *refinement* that is akin to the notion of implication in logical specification languages. (See the paper [3] for a thorough analysis of the connections between specifications given in terms of MTSs and logical specifications in the setting of a modal logic that characterizes refinement.) In an MTS, transitions come in two flavours: the *may* transitions and the *must* transitions, with the

* Research supported by Spanish projects DESAFIOS10 TIN2009-14599-C03-01, TESIS TIN2009-14321-C02-01 and PROMETIDOS S2009/TIC-1465, the project 'Processes and Modal Logics' (project nr. 100048021) of the Icelandic Fund for Research, and the Abel Extraordinary Chair programme within the NILS Mobility Project.

requirement that each must transition is also a may transition. The idea behind the notion of refinement over MTSs is that, in order to implement correctly a specification, an implementation should exhibit all the transitions that are required by the specification (these are the must transitions in the MTS that describes the specification) and may provide the transitions that are allowed by the specification (these are the may transitions in the MTS that describes the specification).

The formalism of modal transition systems is intuitive, has several variants with varying degrees of expressive power and complexity—see, e.g., the survey paper [1]—and has recently been used as a suitable model for the specification of service-oriented applications. In particular, results on the supervisory control (in the sense of Ramadge and Wonham [15]) of systems whose specification is given in that formalism have been presented in, e.g., [4,8].

The very recent development of the notion of *partial bisimulation* in the setting of labelled transition systems (LTSs) presented in [2] has been explicitly motivated by the desire to develop a process-algebraic model within which one can study topics in the field of supervisory control. A partial bisimulation is a variation on the classic notion of bisimulation [13,14] in which two LTSs are only required to fulfil the bisimulation conditions on a subset B of the collection of actions; transitions labelled by actions not in B are treated as in the standard simulation preorder. Intuitively, one may think of the actions in B as corresponding to the uncontrollable events—see [2, page 4]. The aforementioned paper offers a thorough development of the basic theory of partial bisimulation.

Another recent proposal for a simulation-based behavioural relation over LTSs, called the *covariant-contravariant simulation preorder*, has been put forward in [5], and its theory has been investigated further in [6]. This notion of simulation between LTSs is based on considering a partition of their set of actions into three sets: the collection of covariant actions, that of contravariant actions and the set of bivariant actions. Intuitively, one may think of the covariant actions as being under the control of the specification LTS, and transitions with such actions as their label should be simulated by any correct implementation of the specification. On the other hand, the contravariant actions may be considered as being under the control of the implementation (or of the environment) and transitions with such actions as their label should be simulated by the specification. The bivariant actions are treated as in the classic notion of bisimulation.

It is natural to wonder whether there are any relations among these three formalisms. In particular, one may ask oneself whether it is possible to offer mutual translations between specifications given in those state-transition-based models that preserve, and reflect, the appropriate notions of behavioural preorder as well as properties expressed in the modal logics that accompany them—see, e.g., [2,3,6]. The aim of this study is to offer an answer to this question.

In this paper, we study the relationships between refinement over modal transition systems, and the covariant-contravariant simulation and the partial bisimulation preorders over labelled transition systems. We offer mutual trans-

lations between modal transition systems and labelled transition systems that preserve, and reflect, refinement and the covariant-contravariant simulation preorder, as well as the modal properties that can be expressed in the logics that characterize those preorders. We also give a translation from labelled transition systems modulo the partial bisimulation preorder into the same model modulo the covariant-contravariant simulation preorder, together with some evidence that the former model is less expressive than the latter. Finally, in order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, we phrase and study their connections in the context of institutions [9].

The developments in this paper indicate that the formalism of MTSs may be seen as a common ground within which one can embed LTSs modulo the covariant-contravariant simulation preorder or partial bisimilarity. Moreover, there are some interesting, and non-obvious, corollaries that one may infer from the translations we provide. See Section 5, where we use our translations to show, e.g., that checking whether two states in an LTS are related by the covariant-contravariant simulation preorder can always be reduced to an equivalent check in a setting without bivariant actions, and provide a more detailed analysis of the translations. The study of the relative expressive power of different formalisms is, however, an art as well as a science, and may yield different answers depending on the conceptual framework that one adopts for the comparison. For instance, at the level of institutions [9], we provide an institution morphism from the institution corresponding to the theory of MTSs modulo refinement into the institution corresponding to the theory of LTSs modulo the covariant-contravariant simulation preorder. However, we conjecture that there is no institution morphism in the other direction. The work presented in the study opens several interesting avenues for future research, and settling the above conjecture is one of a wealth of research questions we survey in Section 8.

The remainder of the paper is organized as follows. Section 2 is devoted to preliminaries. In particular, in that section, we provide all the necessary background on modal and labelled transition systems, modal refinement and the covariant-contravariant simulation preorder, and the modal logics that characterize those preorders. In Section 3, we show how one can translate LTSs modulo the covariant-contravariant simulation preorder into MTSs modulo refinement. Section 4 presents the converse translation. We discuss the mutual translations between LTSs and MTSs in Section 5. Section 6 offers a translation from LTSs modulo partial bisimilarity into LTSs modulo the covariant-contravariant simulation preorder. In Section 7, we study the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder in the context of institutions. Section 8 concludes the paper and offers a number of directions for future research that we plan to pursue.

The proofs of all the results in the paper and further developments may be found in the full version of this study, which is available at
`http://www.ru.is/faculty/luca/PAPERS/mts-cc.pdf`.

2 Preliminaries

We begin by introducing modal transition systems, with their associated notion of (modal) refinement, and labelled transition systems modulo the covariant-contravariant simulation preorder. We refer the reader to, e.g., [3,11,12] and [5,6] for more information, motivation and examples.

Modal transition systems and refinement

Definition 1. *For a set of actions A, a* modal transition *system (MTS) is a triple $(P, \to_\diamond, \to_\Box)$, where P is a set of states and $\to_\diamond, \to_\Box \subseteq P \times A \times P$ are transition relations such that $\to_\Box \subseteq \to_\diamond$.*

An MTS is image finite *iff the set $\{p' \mid p \xrightarrow{a}_\diamond p'\}$ is finite for each $p \in P$ and $a \in A$.*

The transitions in $\to_\Box$ are called the *must transitions* and those in $\to_\diamond$ are the *may transitions*. In an MTS, each must transition is also a may transition, which intuitively means that any required transition is also allowed.

In what follows, we often identify an MTS, or a transition system of any of the types that we consider in this paper, with its set of states. In case we wish to make clear the 'ambient' transition system in which a state p lives, we write (P, p) to indicate that p is to be viewed as a state in P.

The notion of (modal) refinement $\sqsubseteq$ over MTSs that we now proceed to introduce is based on the idea that if $p \sqsubseteq q$ then q is a 'refinement' of the specification p. In that case, intuitively, q may be obtained from p by possibly turning some of its may transitions into must transitions.

Definition 2. *A relation $R \subseteq P \times Q$ is a* refinement relation *between two modal transition systems if, whenever $p\ R\ q$:*

- *$p \xrightarrow{a}_\Box p'$ implies that there exists some q' such that $q \xrightarrow{a}_\Box q'$ and $p'\ R\ q'$;*
- *$q \xrightarrow{a}_\diamond q'$ implies that there exists some p' such that $p \xrightarrow{a}_\diamond p'$ and $p'\ R\ q'$.*

We write $\sqsubseteq$ for the largest refinement relation.

Example 1. Consider the MTS U over the set of actions A with u as its only state, and transitions $u \xrightarrow{a}_\diamond u$ for each $a \in A$. It is well known, and not hard to see, that $u \sqsubseteq p$ holds for each state p in any MTS over action set A. The state u is often referred to as the *loosest (or universal) specification.*

Definition 3. *Given a set of actions A, the collection of* Boudol-Larsen's modal formulae [3] *is given by the following grammar:*

$$\varphi ::= \bot \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [a]\varphi \mid \langle a \rangle \varphi \qquad (a \in A).$$

The semantics of these formulae with respect to an MTS P and a state $p \in P$ is defined by means of the satisfaction relation $\models$, which is the least relation satisfying the following clauses:

$(P,p) \models \top$.
$(P,p) \models \varphi_1 \wedge \varphi_2$ *if* $(P,p) \models \varphi_1$ *and* $(P,p) \models \varphi_2$.
$(P,p) \models \varphi_1 \vee \varphi_2$ *if* $(P,p) \models \varphi_1$ *or* $(P,p) \models \varphi_2$.
$(P,p) \models [a]\varphi$ *if* $(P,p') \models \varphi$ *for all* $p \stackrel{a}{\rightarrow}_{\diamond} p'$.
$(P,p) \models \langle a\rangle\varphi$ *if* $(P,p') \models \varphi$ *for some* $p \stackrel{a}{\rightarrow}_{\Box} p'$.

For example, the state U from Example 1 satisfies neither the formula $\langle a\rangle\top$ nor the formula $[a]\bot$. Indeed, it is not hard to see that U satisfies a formula φ if, and only if, φ is a tautology.

The following result stems from [3].

Proposition 1. *Let p, q be states in image-finite MTSs over the set of actions A. Then $p \sqsubseteq q$ iff the collection of Boudol-Larsen's modal formulae satisfied by p is included in the collection of formulae satisfied by q.*

Labelled transition systems and covariant-contravariant simulation A labelled transition system (LTS) is just an MTS with $\rightarrow_{\diamond}=\rightarrow_{\Box}$. In what follows, we write $\rightarrow$ for the transition relation in an LTS.

Definition 4. *Let P and Q be two LTSs over the set of actions A, and let $\{A^r, A^l, A^{bi}\}$ be a partition of A*[3]*. An* (A^r, A^l)-simulation *(or just a* covariant-contravariant simulation *when the partition of the set of actions A is understood from the context) between P and Q is a relation $R \subseteq P \times Q$ such that, whenever $p\ R\ q$, we have:*

- *For all $a \in A^r \cup A^{bi}$ and all $p \stackrel{a}{\rightarrow} p'$, there exists some $q \stackrel{a}{\rightarrow} q'$ with $p'\ R\ q'$.*
- *For all $a \in A^l \cup A^{bi}$ and all $q \stackrel{a}{\rightarrow} q'$, there exists some $p \stackrel{a}{\rightarrow} p'$ with $p'\ R\ q'$.*

We will write $p \lesssim_{cc} q$ if there exists a covariant-contravariant simulation R such that $p\ R\ q$.

The actions in the set A^r are sometimes called *covariant*, those in A^l are *contravariant* and the ones in A^{bi} are *bivariant*. When working with covariant-contravariant simulations, we shall sometimes refer to the triple (A^r, A^l, A^{bi}) as the *signature* of the corresponding LTS.

Example 2. Assume that $a \in A^r$ and $b \in A^l$. Consider the LTSs described by the CCS [13] terms $p = a + b$, $q = a$ and $r = b$. Then $r \lesssim_{cc} p \lesssim_{cc} q$, but none of the converse relations holds.

Definition 5. Covariant-contravariant modal logic *has almost the same syntax as the one for modal refinement:*

$$\varphi ::= \bot \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [b]\varphi \mid \langle a\rangle\varphi \qquad (a \in A^r \cup A^{bi}, b \in A^l \cup A^{bi}).$$

The semantics differs for the modal operators, since we interpret formulae over ordinary LTSs:

[3] Note that any of the sets A^r, A^l and A^{bi} may be empty.

$(P,p) \models [b]\varphi$ *if* $(P,p') \models \varphi$ *for all* $p \xrightarrow{b} p'$.
$(P,p) \models \langle a\rangle\varphi$ *if* $(P,p') \models \varphi$ *for some* $p \xrightarrow{a} p'$.

For example, both p and q from Example 2 satisfy the formula $\langle a\rangle\top$, while r does not. On the other hand, q satisfies the formula $[b]\bot$, but neither p nor r do.

The following result stems from [6].

Proposition 2. *Let p, q be states in image-finite LTSs with the same signature. Then $p \lesssim_{cc} q$ iff the collection of covariant-contravariant modal formulae satisfied by p is included in the collection of covariant-contravariant modal formulae satisfied by q.*

3 From covariant-contravariant simulations to modal refinements

We are now ready to begin our study of the connections between MTSs modulo refinement and LTSs modulo the covariant-contravariant simulation preorder. First we show that, perhaps surprisingly, LTSs modulo $\lesssim_{cc}$ may be translated into MTSs modulo $\sqsubseteq$. Such a translation preserves, and reflects, those preorders and the satisfaction of modal formulae.

Definition 6. *Let P be an LTS with the set of actions A partitioned into A^r, A^l, and A^{bi}. The MTS $\mathcal{M}(P)$ is constructed as follows:*

- *The set of actions of $\mathcal{M}(P)$ is A.*
- *The set of states of $\mathcal{M}(P)$ is the same as the one of P plus a new state u.*
- *For each transition $p \xrightarrow{a} p'$ in P, add a may transition $p \xrightarrow{a}_{\diamond} p'$ in $\mathcal{M}(P)$.*
- *For each transition $p \xrightarrow{a} p'$ in P with $a \in A^r \cup A^{bi}$, add a must transition $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$.*
- *For each a in A^r and state p, add the transition $p \xrightarrow{a}_{\diamond} u$ to $\mathcal{M}(P)$, as well as transitions $u \xrightarrow{a}_{\diamond} u$ for each action $a \in A$.*

The following proposition essentially states that the translation $\mathcal{M}$ is correct.

Proposition 3. *Let P and Q be two LTSs with the same signature, and let $p \in P$ and $q \in Q$. Then $(P,p) \lesssim_{cc} (Q,q)$ iff $(\mathcal{M}(P),p) \sqsubseteq (\mathcal{M}(Q),q)$.*

Proof. We prove the two implications separately.

($\Rightarrow$) Assume that R is a covariant-contravariant simulation. We shall prove that $\mathcal{M}(R) = R \cup \{(u,q) \mid q \text{ a state of } \mathcal{M}(Q)\}$ is a refinement.

Suppose that $p\ R\ q$ and $q \xrightarrow{a}_{\diamond} q'$ in $\mathcal{M}(Q)$. By the definition of $\mathcal{M}(Q)$, the transition $q \xrightarrow{a} q'$ is in Q. If $a \in A^l \cup A^{bi}$, since $p\ R\ q$ and R is a covariant-contravariant simulation, we have that $p \xrightarrow{a} p'$ in P for some p' such that $p'\ R\ q'$. By the construction of $\mathcal{M}(P)$, it holds that $p \xrightarrow{a}_{\diamond} p'$ and we are done. If $a \in A^r$, then $p \xrightarrow{a}_{\diamond} u$ and $u\ \mathcal{M}(R)\ q'$, as required.

Assume now that $p\ R\ q$ and $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$. Then $p \xrightarrow{a} p'$ in P with $a \in A^r \cup A^{bi}$. As R is a covariant-contravariant simulation, it follows that $q \xrightarrow{a} q'$

in Q for some q' such that $p'\ R\ q'$. Since $a \in A^r \cup A^{bi}$, there is a must transition $q \xrightarrow{a}_{\Box} q'$ in $\mathcal{M}(Q)$, and we are done. To finish the proof of this implication, recall that, as shown in Example 1, q is a refinement of u for each q.

($\Leftarrow$) Assume that $\mathcal{M}(R)$ is a refinement. We shall prove that R is a covariant-contravariant simulation.

Suppose that $p\ R\ q$ and $q \xrightarrow{a} q'$ in Q with $a \in A^l \cup A^{bi}$. Then $q \xrightarrow{a}_{\diamond} q'$ in $\mathcal{M}(Q)$. Since $\mathcal{M}(R)$ is a refinement, in $\mathcal{M}(P)$ we have that $p \xrightarrow{a}_{\diamond} p'$ for some p' (different from u, because $a \notin A^r$) such that $p'\ R\ q'$. By the construction of $\mathcal{M}(P)$, it follows that $p \xrightarrow{a} p'$ in P and we are done.

Suppose now that $p\ R\ q$ and $p \xrightarrow{a} p'$ in P with $a \in A^r \cup A^{bi}$. Then $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$. Since $\mathcal{M}(R)$ is a refinement, there is some q' (again, different from u) such that $q \xrightarrow{a}_{\Box} q'$ in $\mathcal{M}(Q)$ and $p'\ R\ q'$. By the construction of $\mathcal{M}(Q)$, it follows that $q \xrightarrow{a} q'$ in Q and we are done. □

Definition 7. *Let us extend $\mathcal{M}$ to translate formulae over the modal logic that characterizes the covariant-contravariant simulation preorder to the modal logic for modal transition systems by simply defining $\mathcal{M}(\varphi) = \varphi$.*

Proposition 4. *If P is an LTS and φ is a formula of the logic that characterizes covariant-contravariant simulation, then for each $p \in P$:*

$$(P, p) \models \varphi \iff (\mathcal{M}(P), p) \models \mathcal{M}(\varphi).$$

4 From modal refinements to covariant-contravariant simulations

We now show that MTSs modulo $\sqsubseteq$ may be translated into LTSs modulo $\lesssim_{cc}$. As the one studied in the previous section, our translation preserves, and reflects, those preorders and the satisfaction of modal formulae.

Definition 8. *Let M be an MTS with set of actions A. The LTS $\mathcal{C}(M)$, with signature $A^r = \{\mathrm{cv}(a) \mid a \in A\}$, $A^l = \{\mathrm{ct}(a) \mid a \in A\}$ and $A^{bi} = \emptyset$, is constructed as follows:*

- *The set of states of $\mathcal{C}(M)$ is the same as that of M.*
- *For each transition $p \xrightarrow{a}_{\diamond} p'$ in M, add $p \xrightarrow{\mathrm{ct}(a)} p'$ to $\mathcal{C}(M)$.*
- *For each transition $p \xrightarrow{a}_{\Box} p'$ in M, add $p \xrightarrow{\mathrm{cv}(a)} p'$ to $\mathcal{C}(M)$.*

Observe that the LTSs obtained as a translation of an MTS have the following properties:

1. $A^{bi} = \emptyset$ and
2. there is a bijection $h : A^r \to A^l$ such that if $p \xrightarrow{a} p'$ with $a \in A^r$ then $p \xrightarrow{h(a)} p'$.

The following proposition essentially states that the translation $\mathcal{C}$ is correct.

Proposition 5. *Let P and Q be two MTSs with the same action set, and let $p \in P$ and $q \in Q$. Then $(P,p) \sqsubseteq (Q,q)$ iff $(\mathcal{C}(P),p) \lesssim_{cc} (\mathcal{C}(Q),q)$.*

Proof. We prove the two implications separately.

($\Rightarrow$) Assume that $p\ R\ q$ for some refinement R. If $p \xrightarrow{\mathrm{cv}(a)} p'$ in $\mathcal{C}(P)$ then, by construction, $p \xrightarrow{a}_{\Box} p'$ in P. Since R is a refinement, there is some q' in Q with $q \xrightarrow{a}_{\Box} q'$ and $p'\ R\ q'$. Since $q \xrightarrow{\mathrm{cv}(a)} q'$ is in $\mathcal{C}(Q)$ by construction, we are done. Now, assume that $q \xrightarrow{\mathrm{ct}(a)} q'$ in $\mathcal{C}(Q)$. Then $q \xrightarrow{a}_{\diamond} q'$ in Q and, since R is a refinement, $p \xrightarrow{a}_{\diamond} p'$ in P for some p' with $p'\ R\ q'$. By construction, $p \xrightarrow{\mathrm{ct}(a)} p'$ is in $\mathcal{C}(P)$ and we are done.

($\Leftarrow$) Assume that $p\ R\ q$ for some covariant-contravariant simulation R. If $q \xrightarrow{a}_{\diamond} q'$ in Q then $q \xrightarrow{\mathrm{ct}(a)} q'$ in $\mathcal{C}(Q)$ and, since R is a covariant-contravariant simulation, $p \xrightarrow{\mathrm{ct}(a)} p'$ for some p' in $\mathcal{C}(P)$ such that $p'Rq'$; hence $p \xrightarrow{a}_{\diamond} p'$ in P as required. Now, if $p \xrightarrow{a}_{\Box} p'$ in P then $p \xrightarrow{\mathrm{cv}(a)} p'$ in $\mathcal{C}(P)$. Since R is a covariant-contravariant simulation, there is some q' in $\mathcal{C}(Q)$ with $q \xrightarrow{\mathrm{cv}(a)} q'$ and $p'\ R\ q'$, and therefore $q \xrightarrow{a}_{\Box} q'$ in Q. □

Definition 9. *Let us extend $\mathcal{C}$ to translate formulae over the modal logic for modal transition systems with set of actions A to the modal logic that characterizes covariant-contravariant simulation with signature $A^r = \{\mathrm{cv}(a) \mid a \in A\}$, $A^l = \{\mathrm{ct}(a) \mid a \in A\}$ and $A^{bi} = \emptyset$.*

- $\mathcal{C}(\bot) = \bot$.
- $\mathcal{C}(\top) = \top$.
- $\mathcal{C}(\varphi \wedge \psi) = \mathcal{C}(\varphi) \wedge \mathcal{C}(\psi)$.
- $\mathcal{C}(\varphi \vee \psi) = \mathcal{C}(\varphi) \vee \mathcal{C}(\psi)$.
- $\mathcal{C}(\langle a \rangle \varphi) = \langle \mathrm{cv}(a) \rangle \mathcal{C}(\varphi)$.
- $\mathcal{C}([a]\varphi) = [\mathrm{ct}(a)]\mathcal{C}(\varphi)$.

Proposition 6. *If P is an MTS and φ a modal formula, then for each $p \in P$:*

$$(P,p) \models \varphi \iff (\mathcal{C}(P),p) \models \mathcal{C}(\varphi).$$

5 Discussion of the translations

In Sections 3–4, we saw that it is possible to translate back and forth between the world of LTSs modulo the covariant-contravariant simulation preorder and MTSs modulo refinement. The translations we have presented preserve, and reflect, the preorders and the relevant modal formulae. There are, however, some interesting, and non-obvious, corollaries that one may infer from the translations.

To begin with, assume that P and Q are two LTSs with the same signature with $A^{bi} \neq \emptyset$. Let $p \in P$ and $q \in Q$ be such that $(P,p) \lesssim_{cc} (Q,q)$. By Proposition 3, we know that this holds exactly when $(\mathcal{M}(P),p) \sqsubseteq (\mathcal{M}(Q),q)$. Using Proposition 5, we therefore have that checking whether $(P,p) \lesssim_{cc} (Q,q)$ is

equivalent to verifying whether $(\mathcal{C}(\mathcal{M}(P)), p) \lesssim_{cc} (\mathcal{C}(\mathcal{M}(Q)), q)$. Note now that A^{bi} is empty in the signature for the LTSs $\mathcal{C}(\mathcal{M}(P))$ and $\mathcal{C}(\mathcal{M}(Q))$. Therefore checking whether two states are related by the covariant-contravariant simulation preorder can always be reduced to an equivalent check in a setting without bivariant actions.

It is also natural to wonder whether there is any relation between a state p in an LTS P and the equally-named state in $\mathcal{C}(\mathcal{M}(P))$. Similarly, one may wonder whether there is any relation between a state p in an MTS P and the equally-named state in $\mathcal{M}(\mathcal{C}(P))$. In both cases, we are faced with the difficulty that the transition systems resulting from the compositions of the translations are over actions of the form $\{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\}$ whereas the original systems had transitions labelled by actions in A. In order to overcome this difficulty, let $\rho : \{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\} \to A$ be the renaming that, for each $a \in A$, maps both $\mathrm{cv}(a)$ and $\mathrm{ct}(a)$ to a. For any transition system P over the set of actions $\{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\}$, we write $\rho(P)$ for the transition system that is obtained from P by renaming the label of each transition in P as indicated by ρ.

Proposition 7.

1. *Let P be an MTS and let $p \in P$. Then $(\rho(\mathcal{M}(\mathcal{C}(P))), p) \sqsubseteq (P, p)$.*
2. *Let P be an LTS and let $p \in P$. Then $(P, p) \lesssim_{cc} (\rho(\mathcal{C}(\mathcal{M}(P))), p)$.*
3. *In general, $(P, p) \sqsubseteq (\rho(\mathcal{M}(\mathcal{C}(P))), p)$ does not hold for an MTS P and a state $p \in P$, nor does $(\rho(\mathcal{C}(\mathcal{M}(P))), p) \lesssim_{cc} (P, p)$ for an LTS P and a state $p \in P$.*

Definition 10. *Let P be an LTS with the set of actions partitioned into A^r and A^l. The LTS $\overline{P}$ is obtained from P by renaming every $a \in A^r$ as $\mathrm{cv}(a)$ and every $a \in A^l$ as $\mathrm{ct}(a)$.*

Proposition 8. *Let P be an LTS over a set of actions $A^r \cup A^l$ and let Q be an MTS over the same actions. Then the following statements hold.*

1. *If a relation R is a covariant-contravariant simulation between $\overline{P}$ and $\mathcal{C}(Q)$ then R is a refinement between $\mathcal{M}(P)$ and Q.*
2. *If $(\overline{P}, p) \lesssim_{cc} (\mathcal{C}(Q), q)$ then $(\mathcal{M}(P), p) \sqsubseteq (Q, q)$, for all states $p \in P$ and $q \in Q$.*
3. *The converse implication of the above statement fails.*

6 Partial bisimulation

The partial bisimulation preorder has been recently proposed in [2] as a suitable behavioural relation over LTSs for studying the theory of supervisory control [15] in a concurrency-theoretic framework. Formally, the notion of partial bisimulation is defined over LTSs with a set of actions A and a so-called *bisimulation set* $B \subseteq A$. The LTSs considered in [2] also include a termination predicate $\downarrow$ over states. For the sake of simplicity, since its role is orthogonal to our aims in this paper, instead of extending MTSs and their refinements or covariant-contravariant simulations with such a predicate, we simply omit it in what follows.

Definition 11. *A* partial bisimulation with bisimulation set B *between two LTSs P and Q is a relation $R \subseteq P \times Q$ such that, whenever $p\ R\ q$:*

- *For all $a \in A$, if $p \xrightarrow{a} p'$ then there exists some $q \xrightarrow{a} q'$ with $p'\ R\ q'$.*
- *For all $b \in B$, if $q \xrightarrow{b} q'$ then there exists some $p \xrightarrow{b} p'$ with $p'\ R\ q'$.*

We write $p \lesssim_B q$ if $p\ R\ q$ for some partial bisimulation with bisimulation set B.

It is easy to see that partial bisimulation with bisimulation set B is a particular case of covariant-contravariant simulation.

Proposition 9. *Let P be an LTS. A relation R is a partial bisimulation with bisimulation set B iff it is a covariant-contravariant simulation when the LTS P has signature $A^r = A \setminus B$, $A^l = \emptyset$ and $A^{bi} = B$. Therefore, $p \lesssim_B q$ iff $p \lesssim_{cc} q$ with respect to that partition of A, for each $p, q \in P$.*

As a corollary of the above proposition, we immediately obtain the following result, to the effect that, instead of the modal logic used in [2] to characterize the partial bisimulation preorder with bisimulation set B, one can use the simpler, negation-free logic for the covariant-contravariant simulation preorder.

Corollary 1. *Let p, q be states in some image-finite LTS. Then $p \lesssim_B q$ iff the collection of formulae in Definition 5 over signature $A^r = A \setminus B$, $A^l = \emptyset$ and $A^{bi} = B$ satisfied by p is included in the collection of formulae satisfied by q.*

Note also that, as a corollary of Proposition 9, the translations of LTSs and formulae defined in Section 3 can be applied to embed LTSs modulo the partial bisimulation preorder into modal transition systems modulo refinement. In this case, however, there is an easier transformation that does not require the extra state u.

Definition 12. *Let P be an LTS over a set of actions A with a bisimulation set $B \subseteq A$. The MTS $\mathcal{N}(P)$ is constructed as follows:*

- *The set of states is that of P.*
- *For each transition $p \xrightarrow{a} p'$ in P, add a may transition $p \xrightarrow{a}_\diamond p'$ in $\mathcal{N}(P)$.*
- *For each transition $p \xrightarrow{b} p'$ in P with $b \in B$, add a must transition $p \xrightarrow{b}_\Box p'$ in $\mathcal{N}(P)$.*

Proposition 10. *R is a partial bisimulation with bisimulation set B between P and Q iff R^{-1} is a refinement between $\mathcal{N}(Q)$ and $\mathcal{N}(P)$.*

Proof. ($\Rightarrow$) Assume that R is a partial bisimulation with bisimulation set B and suppose that $q\ R^{-1}\ p$. If $p \xrightarrow{a}_\diamond p'$ in $\mathcal{N}(P)$ then $p \xrightarrow{a} p'$ in P. Since R is a partial bisimulation, there is some $q \xrightarrow{a} q'$ in Q with $p'\ R\ q'$ and, by construction, $q \xrightarrow{a}_\diamond q'$ in $\mathcal{N}(Q)$ with $q'\ R^{-1}\ p'$. Now, if $q \xrightarrow{a}_\Box q'$ in $\mathcal{N}(Q)$ then $q \xrightarrow{a} q'$ in Q with $a \in B$. Since R is a partial bisimulation and $p\ R\ q$, there is some $p \xrightarrow{a} p'$ in P with $p'\ R\ q'$ and hence $p \xrightarrow{a}_\Box p'$ in $\mathcal{N}(P)$, as required.

($\Leftarrow$) Analogous. □

Remark 1. In the special case $B = \emptyset$, the partial bisimulation preorder is just the standard simulation preorder. Therefore, letting 0 denote a one-state LTS with no transitions, $0 \lesssim_B p$ for each state p in any LTS P. Since $B = \emptyset$, all the modal transition systems $\mathcal{N}(P)$ that result from the translation of an LTS P will have no must transitions; for such modal transition systems, $\mathcal{N}(P) \sqsubseteq 0$ always holds. Indeed, in that case $\sqsubseteq$ coincides with the inverse of the simulation preorder over MTSs.

The drawback of the direct transformation presented in Definition 12, as compared to that in Section 3, is that it does not preserve the satisfiability of modal formulae. The problem lies in the fact that, while the existential modality $\langle a \rangle$ allows any transition with $a \in A$ in the partial bisimulation framework, it requires a must transition in the setting of MTSs.

As we have seen, it is easy to express partial bisimulations as a special case of covariant-contravariant simulations. It is therefore natural to wonder whether the converse also holds. We shall present some indications that the partial bisimulation framework is strictly less expressive than both modal refinements and covariant-contravariant simulations.

Let us assume, by way of example, that the set of actions A is partitioned into $A^r = \{a\}$ and $A^l = \{b\}$—so the set of bivariant actions is empty. In this setting, there cannot be a translation $\mathcal{T}$ from LTSs modulo $\lesssim_{cc}$ into LTSs modulo $\lesssim_B$ that satisfies the following natural conditions (by abuse of notation, we identify an LTS P with a specific state p):

1. For all p and q, $p \lesssim_{cc} q \iff \mathcal{T}(p) \lesssim_B \mathcal{T}(q)$.
2. $\mathcal{T}$ is a homomorphism with respect to $+$, that is, $\mathcal{T}(p + q) = \mathcal{T}(p) + T(q)$, where $+$ denotes the standard notion of nondeterministic composition of LTSs from CCS [13]. (Intuitively, this compositionality requirement states that the translation is based on 'local information'.)
3. There is an n such that $\mathcal{T}(b^n)$ is not simulation equivalent to $\mathcal{T}(0)$, where b^n denotes an LTS consisting of n consecutive b-labelled transitions.

Indeed, observe that, by condition 2,

$$\mathcal{T}(p) = \mathcal{T}(p + 0) = \mathcal{T}(p) + \mathcal{T}(0) \quad \text{for each } p,$$

and therefore $\mathcal{T}(p) + \mathcal{T}(0) \lesssim_B \mathcal{T}(p)$. This means that $\mathcal{T}(0) \lesssim \mathcal{T}(p)$ for each p, where $\lesssim$ is the simulation preorder. In particular, $\mathcal{T}(0) \lesssim \mathcal{T}(\bot)$ where $\bot$ is the process consisting of a b-labelled loop with one state, which is the least element with respect to $\lesssim_{cc}$.

Note now that $\bot \lesssim_{cc} b^{n+1} \lesssim_{cc} b^n \lesssim_{cc} 0$ for each $n > 0$. Therefore, by condition 1,

$$\mathcal{T}(\bot) \lesssim_B \mathcal{T}(b^{n+1}) \lesssim_B \mathcal{T}(b^n) \lesssim_B \mathcal{T}(0) \quad \text{for each } n > 0.$$

Hence,

$$\mathcal{T}(\bot) \lesssim \mathcal{T}(b^n) \lesssim \mathcal{T}(0) \lesssim \mathcal{T}(\bot) \quad \text{for each } n > 0.$$

This yields that, for each $n > 0$, $\mathcal{T}(b^n)$ is simulation equivalent to $\mathcal{T}(0)$, which contradicts condition 3. (Note that we have only used the soundness of the transformation $\mathcal{T}$.)

This indicates strongly that any $\mathcal{T}$ that is compositional with respect to $+$ and is sound, in the sense of condition 1, would have to be very odd indeed, if it exists at all. Modulo simulation equivalence, such a translation would have to conflate a non-well-founded descending chain of LTSs into a point modulo simulation equivalence.

We end this section with a companion result.

Proposition 11. *Assume that $a \in A^r$ and $b \in A^l$. Suppose furthermore that $B = \emptyset$. Then there is no translation $\mathcal{T}$ from LTSs modulo $\lesssim_{cc}$ into LTSs modulo $\lesssim_B$ that satisfies conditions 1 and 2 above.*

7 Institutions and institution morphisms

In order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, we will now study their connections at a more abstract level in the context of institutions [9]. When compared at the level of institutions it turns out that the correspondence between these models is, in a sense, not one-to-one.

Definition 13. *The institution $\mathcal{I}_{cc} = (\mathbf{Sign}_{cc}, sen_{cc}, \mathbf{Mod}_{cc}, \models_{cc})$, associated to the logic for the covariant-contravariant simulation preorder, is defined as follows.*

- $\mathbf{Sign}_{cc}$ *has as objects triples (A, B, C) of pairwise disjoint sets and morphisms $f : A \cup B \cup C \longrightarrow A' \cup B' \cup C'$ with $f(A) \subseteq A'$, $f(B) \subseteq B'$, and $f(C) \subseteq C'$.*
- $sen_{cc}(A, B, C)$ *is the set of formulae in the logic characterizing the covariant-contravariant simulation preorder, with A the set of covariant actions, B the set of contravariant actions, and C the set of bivariant actions. $sen(f)(\varphi)$ is obtained from φ by replacing each action a with $f(a)$.*
- $\mathbf{Mod}_{cc}(A, B)$ *is the category of LTS over the set of actions $A \cup B \cup C$, with a distinguished state; a morphism from (P, p) to (Q, q) is a covariant-contravariant simulation R such that $(p, q) \in R$.*
  Now, if $f : A \cup B \cup C \longrightarrow A' \cup B' \cup C'$, then

$$\mathbf{Mod}_{cc}(f) : \mathbf{Mod}_{cc}(A', B', C') \longrightarrow \mathbf{Mod}_{cc}(A, B, C)$$

  maps P to $P|_f$ and $R : P \longrightarrow Q$ to $R_f : P|_f \longrightarrow Q|_f$, where:
  - *The set of states of $P|_f$ is the same as that of P, and the distinguished state remains the same.*
  - *$p \xrightarrow{a} p'$ in $P|_f$ if $p \xrightarrow{f(a)} p'$ in P.*
  - *$R|_f$ coincides with R.*

– $(P, s) \models_{cc} \varphi$ *if* $(P, s) \models \varphi$ *using the notion of satisfaction associated to the logic for the covariant-contravariant simulation preorder given in Definition 5.*

Proposition 12. *$\mathcal{I}_{cc}$ is an institution.*

Definition 14. *The institution $\mathcal{I}_{mts} = (\mathbf{Sign}_{mts}, sen_{mts}, \mathbf{Mod}_{mts}, \models_{mts})$, associated to the logic for refinement over modal transition systems, is defined as follows.*

– $\mathbf{Sign}_{mts}$ *is the category of sets.*
– $sen_{mts}(A)$ *is the set of formulae over A in the logic presented in Definition 3. The formula $sen_{mts}(f)(\varphi)$ is obtained from φ by replacing each action a with $f(a)$.*
– $\mathbf{Mod}_{mts}(A)$ *is the category of MTSs over the set of labels A, with a distinguished state. A morphism from (M, m) to (N, n) is a refinement R such that $(m, n) \in R$.*
If $f : A \longrightarrow B$ in $\mathbf{Sign}_{mts}$, then $\mathbf{Mod}_{mts}(f) : \mathbf{Mod}_{mts}(B) \longrightarrow \mathbf{Mod}_{mts}(A)$ maps an MTS M to $M|_f$ and a morphism R to $R|_f$, where:
  - *$M|_f$ has the same set of states as M and the same distinguished state.*
  - *$p \xrightarrow{a}_{\diamond} p'$ in $M|_f$ if $p \xrightarrow{f(a)}_{\diamond} p'$ in M.*
  - *$p \xrightarrow{a}_{\Box} p'$ in $M|_f$ if $p \xrightarrow{f(a)}_{\Box} p'$ in M.*
  - *$R|_f$ coincides with R.*
– $\models_{mts}$ *is the notion of satisfaction presented in Definition 3.*

Proposition 13. *$\mathcal{I}_{mts}$ is an institution.*

As the following result shows, one can translate $\mathcal{I}_{mts}$ into $\mathcal{I}_{cc}$ using an institution morphism. (The intuition for institution morphisms is that they are truth preserving translations from one logical system into another.)

Proposition 14. *$(\Phi, \alpha, \beta) : \mathcal{I}_{mts} \longrightarrow \mathcal{I}_{cc}$ is an institution morphism, where:*

– *$\Phi : \mathbf{Sign}_{mts} \longrightarrow \mathbf{Sign}_{cc}$ maps A to the triple $(\mathrm{cv}(A), \mathrm{ct}(A), \emptyset)$, with:*
  - *$\mathrm{cv}(A) = \{\mathrm{cv}(a) \mid a \in A\}$ and*
  - *$\mathrm{ct}(A) = \{\mathrm{ct}(a) \mid a \in A\}$.*

  For $f : A \longrightarrow B$, we define $\Phi(f)(\mathrm{cv}(a)) = \mathrm{cv}(f(a))$ and $\Phi(f)(\mathrm{ct}(a)) = \mathrm{ct}(f(a))$.
– *The natural transformation $\alpha : sen_{cc} \circ \Phi \Rightarrow sen_{mts}$ translates a formula φ in $sen_{cc}(\mathrm{cv}(A), \mathrm{ct}(A), \emptyset)$ as follows:*
  - *$\alpha(\top) = \top$, $\alpha(\bot) = \bot$.*
  - *$\alpha(\varphi_1 \wedge \varphi_2) = \alpha(\varphi_1) \wedge \alpha(\varphi_2)$.*
  - *$\alpha(\varphi_1 \vee \varphi_2) = \alpha(\varphi_1) \vee \alpha(\varphi_2)$.*
  - *$\alpha(\langle \mathrm{cv}(a) \rangle \varphi) = \langle a \rangle \alpha(\varphi)$.*
  - *$\alpha([\mathrm{ct}(a)]\varphi) = [a]\alpha(\varphi)$.*
– *The natural transformation $\beta : \mathbf{Mod}_{mts} \Rightarrow \mathbf{Mod}_{cc} \circ \Phi$ maps an MTS (M, s) in $\mathbf{Mod}_{mts}(A)$ to $(\mathcal{C}(M), s)$, and a morphism R to itself.*

The import of the above result is that MTSs modulo refinement and its accompanying modal logic can be 'translated in a truth preserving fashion' into LTSs modulo the covariant-contravariant simulation preorder and its companion modal logic. It is natural to ask oneself whether one can consider $\mathcal{I}_{mts}$ a 'subinstitution' of $\mathcal{I}_{cc}$. There are several related notions of subinstitution that have in common the requirement that the functor β, which is used to translate the models between the institutions, is an equivalence of categories.

Recall that an object in a category is weakly final if any other object has at least one arrow into it.

Proposition 15. $\mathbf{Mod}_{cc}(A, B, \emptyset)$ *has weakly final objects but* $\mathbf{Mod}_{mts}(A)$ *does not.*

In other words, in the absence of bivariant actions, there is a universal implementation in the setting of LTSs modulo the covariant-contravariant simulation preorder. Within that framework, there is also a universal specification, namely the LTS (I, s) where I is the LTS with a single state s and transitions $s \xrightarrow{b} s$ for every $b \in B$. On the other hand, there is a universal specification with respect to modal refinements, namely the MTS U from Example 1, but no universal implementation.

Proposition 16. *There is no embedding from* $\mathcal{I}_{mts}$ *into* $\mathcal{I}_{cc}$.

A natural question to ask is whether there is an embedding from $\mathcal{I}_{cc}$ into $\mathcal{I}_{mts}$. The following proposition answers this question negatively.

Proposition 17. *There exists no embedding from* $\mathcal{I}_{cc}$ *into* $\mathcal{I}_{mts}$.

We conjecture that there is not even an institution morphism from $\mathcal{I}_{cc}$ to $\mathcal{I}_{mts}$. (Compare with Proposition 14.)

8 Conclusions and future work

In this paper we have studied the relationships between three notions of behavioural preorders that have been proposed in the literature: refinement over modal transition systems, and the covariant-contravariant simulation and the partial bisimulation preorders over labelled transition systems. We have provided mutual translations between modal transition systems and labelled transition systems that preserve, and reflect, refinement and the covariant-contravariant simulation preorder, as well as the the modal properties that can be expressed in the logics that characterize those preorders. We have also offered a translation from labelled transition systems modulo the partial bisimulation preorder into the same model modulo the covariant-contravariant simulation preorder, together with some evidence that the former model is less expressive than the latter. Finally, in order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the

covariant-contravariant simulation preorder, we have also phrased and studied their connections in the context of institutions.

The work presented in the study opens several interesting avenues for future research. Here we limit ourselves to mentioning a few research directions that we plan to pursue in future work.

First of all, it would be interesting to study the relationships between the LTS-based models we have considered in this article and variations on the MTS model surveyed in, for instance, [1]. In particular, the third author recently contributed in [7] to the comparison of several refinement settings, including modal and mixed transition systems. The developments in that paper offer a different approach to the comparison and application of the formalisms studied in this article.

In [6], three of the authors gave a ground-complete axiomatization of the covariant-contravariant simulation preorder over the language BCCS [13]. It would be interesting to see whether the translations between MTSs and LTSs we have provided in this paper can be used to lift that axiomatization result, as well as results on the nonexistence of finite (in)equational axiomatizations, to the setting of modal transition systems modulo refinement, using the BCCS-like syntax for MTSs given in [3]. We also intend to study whether our translations can be used to obtain characteristic-formula constructions [3,10,16] for one model from extant results on the existence of characteristic formulae for the other.

The existence of characteristic formulae allows one to reduce checking the existence of a behavioural relation between two processes to a model checking question. Conversely, the main result from [3] offers a complete characterization of the model checking questions of the form $(M, m) \models \varphi$, where M is an MTS and φ is a formula in the logic for MTSs considered in this paper, that can be reduced to checking for the existence of a refinement between (M_φ, m_φ) and (M, m), where (M_φ, m_φ) is an MTS with a distinguished state that 'graphically represents' the formula φ. In future work, we plan to offer a characterization of the logical specifications that can be 'graphically represented' by LTSs modulo the covariant-contravariant simulation preorder and partial bisimilarity. Such characterizations may shed further light on the relative expressive power of the two formalisms and may give further evidence of the fact that LTSs modulo the covariant-contravariant simulation preorder are, in some suitable formal sense, more expressive than LTSs modulo partial bisimilarity.

From the theoretical point of view, it would also be satisfying to settle our conjecture that there is no institution morphism from $\mathcal{I}_{cc}$ to $\mathcal{I}_{mts}$.

Last, but not least, the development of the notion of partial bisimulation in [2] has been motivated by the desire to develop a process-algebraic model within which one can study topics in the field of *supervisory control* [15]. Recently, MTSs have been used as a suitable model for the specification of service-oriented applications, and results on the supervisory control of systems whose specification is given in that formalism have been presented in, e.g., [4,8]. It is a very interesting area for future research to study whether the mutual translations between MTSs modulo refinement and LTSs modulo the covariant-contravariant simulation pre-

order can be used to transfer results on supervisory control from MTSs to LTSs. One may also wish to investigate directly the adaptation of the supervisory control theory of Ramadge and Wonham to the enforcement of specifications given in terms of LTSs modulo the covariant-contravariant simulation preorder.

References

1. A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wąsowski. 20 years of modal and mixed specifications. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 95:94–129, 2008.
2. J. Baeten, D. van Beek, B. Luttik, J. Markovski, and J. Rooda. Partial bisimulation. SE Report 2010-04, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology, 2010.
3. G. Boudol and K. G. Larsen. Graphical versus logical specifications. *Theoretical Comput. Sci.*, 106(1):3–20, 1992.
4. P. Darondeau, J. Dubreil, and H. Marchand. Supervisory control for modal specifications of services. In *Proceedings of WODES 2010, August 30–September 1, 2010, Berlin, Germany*, 2010. To appear.
5. I. Fábregas, D. de Frutos-Escrig, and M. Palomino. Non-strongly stable orders also define interesting simulation relations. In *Proceedings of CALCO 2009*, volume 5728 of *Lecture Notes in Computer Science*, pages 221–235. Springer-Verlag, 2009.
6. I. Fábregas, D. de Frutos-Escrig, and M. Palomino. Logics for contravariant simulations. In *Proceedings of FORTE 2010*, volume 6117 of *Lecture Notes in Computer Science*, pages 224–231. Springer-Verlag, 2010.
7. H. Fecher, D. de Frutos-Escrig, G. Lüttgen, and H. Schmidt. On the expressiveness of refinement settings. In *Proceedings of FSEN 2009*, volume 5961 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, 2009.
8. G. Feuillade and S. Pinchinat. Modal specifications for the control theory of discrete event systems. *Discrete Event Dynamical Systems*, 17:211–232, 2007.
9. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.
10. S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68(1–3):125–145, 1986.
11. K. G. Larsen. Modal specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer-Verlag, 1989.
12. K. G. Larsen and B. Thomsen. A modal process logic. In *Proceedings* 3^{th} *Annual Symposium on Logic in Computer Science,* Edinburgh, pages 203–210. IEEE Computer Society Press, 1988.
13. R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.
14. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *5th GI Conference,* Karlsruhe, Germany, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.
15. P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25:206–230, 1987.
16. B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.

Graphical representation of covariant-contravariant modal formulae

Luca Aceto Anna Ingólfsdóttir
ICE-TCS, School of Computer Science
Reykjavik University*
Iceland

Ignacio Fábregas David de Frutos Escrig Miguel Palomino
Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid†
Spain

Covariant-contravariant simulation is a combination of standard (covariant) simulation, its contravariant counterpart and bisimulation. We have previously studied its logical characterization by means of the covariant-contravariant modal logic. Moreover, we have investigated the relationships between this model and that of modal transition systems, where two kinds of transitions (the so-called may and must transitions) were combined in order to obtain a simple framework to express a notion of refinement over state-transition models. In a classic paper, Boudol and Larsen established a precise connection between the *graphical* approach, by means of modal transition systems, and the *logical* approach, based on Hennessy-Milner logic without negation, to system specification. They obtained a (graphical) representation theorem proving that a formula can be represented by a term if, and only if, it is consistent and prime. We show in this paper that the formulae from the covariant-contravariant modal logic that admit a "graphical" representation by means of processes, modulo the covariant-contravariant simulation preorder, are also the consistent and prime ones. In order to obtain the desired graphical representation result, we first restrict ourselves to the case of covariant-contravariant systems without bivariant actions. Bivariant actions can be incorporated later by means of an encoding that splits each bivariant action into its covariant and its contravariant parts.

1 Introduction

Modal transition systems (MTSs) were introduced in [9, 10] as a model of reactive computation based on states and transitions that naturally supports a notion of *refinement*. This is connected with the use of Hennessy-Milner Logic without negation as a specification language: a specification describes the collection of (good) properties that any implementation has to fulfil. More generally, a process p is considered to be better than q if the set of formulae satisfied by q is included in the set of formulae satisfied by p. The tight connections between these two ways of expressing the notions of specification and refinement were studied in [4]. There the authors talked about "graphical" representation (by means of one or several MTSs) of logical specifications, and completely characterized the collection of logical specification that can be "graphically represented". These are the so-called prime, consistent formulae.

There are two types of modal operators in Hennessy-Milner Logic: $\langle a\rangle$ and $[a]$, for each action a. Intuitively, a formula $\langle a\rangle\varphi$ indicates that it must be possible to execute a and reach a state that satisfies

*Research supported by the project 'Processes and Modal Logics' (project nr. 100048021) of the Icelandic Research Fund, and the Abel Extraordinary Chair programme within the NILS Mobility Project.
†Research supported by Spanish projects DESAFIOS10 TIN2009-14599-C03-01, TESIS TIN2009-14321-C02-01 and PROMETIDOS S2009/TIC-1465

B. Luttik and F. D. Valencia (Eds.): 18th International Workshop on Expressiveness in Concurrency (EXPRESS 2011)
EPTCS 64, 2011, pp. 1–15, doi:10.4204/EPTCS.64.1

φ, while $[a]\varphi$ imposes that this will happen after any execution of a from the current state. It is well known that these two operators reflect the duality $\exists$-$\forall$, so that any process satisfying a $\langle a\rangle\varphi$ formula *must* include some a-labelled transition reaching a state satisfying φ, whereas the constraint expressed by a $[a]\varphi$ formula is better understood in a negative way: a process satisfying it *may not* contain an a-labelled transition reaching a state that does not satisfy φ. In particular, the formula $[a]\bot$ indicates that a process cannot execute a in its initial state, and therefore, using these formulae, we can limit the set of actions offered at any state.

In order to reflect these two kinds of constraints at the "operational" level, MTSs contain two kinds of transitions: the *may* transitions and the *must* transitions. Then we can use MTSs both as specifications or as implementations, and the notion of refinement imposes that, in order to implement correctly a specification, an implementation should exhibit all the *must* transitions in the MTS that describes the specification and may not include any transition that is not allowed by the specification: we cannot add any new *may* transition, although those in the specification could either disappear, be preserved or turned into *must* transitions. The relation between *may* and *must* is reflected in the formal definition of MTSs by requiring that each must transition is also a may transition.

The conditions defining the notion of refinement between MTSs obviously resemble those defining simulation and bisimulation. For may transitions we have a contravariant simulation condition, expressing the fact that no new (non-allowed) *may* transition can appear when refining a specification. Since we impose that *must* transitions induce the corresponding *may* transitions, we could think that they are related in a "bisimulation-like" style. However, this is not the case since the contravariant simulation condition imposed on the may part can be covered by a *may* transition without *must* counterpart. In fact, this is crucial in order to capture the principle that a *may* transition can be refined by a *must* transition.

Some of the authors of this paper thought that a more direct combination of simulation and bisimulation conditions could capture in a more flexible way all the ideas on which the specification of systems by means of modal systems and modal logics is based, and we looked for the clearest and most general framework to express those modal constraints. We found that covariant-contravariant systems (sometimes abbreviated to cc-systems) are a possible answer to this quest, combining pure (covariant) simulation, its contravariant counterpart and bisimulation.

We started the study of *covariant-contravariant simulation* in [5], and the modal logic characterizing it was presented in [7]. (In what follows, we refer to this logic as cc-modal logic.) In the most general case, we consider a partition of the set of actions into three sets: the collection of covariant actions, that of contravariant actions, and the set of bivariant actions. Intuitively, one may think of the covariant actions as being under the control of the specification LTS, and transitions with such actions as their label should be simulated by any correct implementation of the specification. On the other hand, the contravariant actions may be considered as being under the control of the implementation (or of the environment) and transitions with such actions as their label should be simulated by the specification. The bivariant actions are treated as in the classic notion of bisimulation.

We will see in this paper that, as in the MTS setting, the consistent and prime formulae from the cc-modal logic are exactly those that admit a "graphical" representation by means of processes modulo the covariant-contravariant simulation preorder. Moreover, each formula in the cc-modal logic can be represented "graphically" by a (possibly empty) finite set of processes.

The proofs of these representation results are inspired by the developments in [4]. There are, however, subtle differences because, in covariant-contravariant systems, each action has a single modality (covariant, contravariant, bivariant), while in MTSs we can combine both *may* and *must* transitions.

In fact, in order to obtain the desired graphical representation, for technical reasons we first restrict ourselves to the case of covariant-contravariant systems without bivariant actions. The reason that justi-

fies this constraint is that bivariant actions cannot be approximated in a non-trivial way (either we have one of them as itself, or we do not have it at all). Instead, covariant and contravariant actions behave in a more flexible way and we can obtain the desired characterization result by following the lead of the work done for MTSs.

Then we observe that bivariant actions can be seen as the combination of a covariant and a contravariant action. In fact, this also corresponds with the idea used in [1] when relating MTSs and cc-systems. Indeed, the constraint imposed on *must* transitions in MTSs, where they should always be accompanied by their *may* counterparts, tells us somehow that they have a "nearly" bivariant behaviour. (To be more precise, they are first covariant, but they are also "semi"-contravariant because when comparing two processes p and q, any *must* transition in q should fit with either a corresponding *must* transition in p, or at least with a *may* transition there.)

We could say that the very recent development of the notion of *partial bisimulation* in the setting of labelled transition systems (LTSs) presented in [3] has completed the spectrum of modal simulations. Partial bisimulation combines plain bisimulation [14, 15] and simulation, also by means of a partition of the set of actions. For the actions in the distinguished set B we have bisimulation-like conditions, while for the others we only impose simulation. Note that, instead, *may* transitions in MTSs corresponded to contravariant simulation conditions, and therefore, partial bisimulation can be seen as a dual of MTSs, and covariant-contravariant systems (cc-systems) as a unifying framework where we can combine the refinement ideas in the theory of MTSs with the explicit consideration of the constraints imposed by the environment, which is possible when partial bisimulation is used. Once we know that the formulae from the modal logic for cc-systems also afford a graphical representation, we will be able to integrate the logical formulae into the development of systems using any of the models discussed above.

The remainder of the paper is organized as follows. Section 2 is devoted to the necessary background on covariant-contravariant simulations, whereas in Section 3 we summarize the results on covariant-contravariant modal formulae. In Section 4 we develop the study of the graphical representation of cc-modal formulae for processes without bivariant actions. Afterwards, in Section 5, we show how we can work with cc-systems with bivariant actions. Finally, Section 6 concludes the paper and describes some future research that we plan to pursue.

2 Covariant-contravariant systems

We start the technical part of the paper by defining the covariant-contravariant simulation semantics for processes. Our semantics is defined over *Labelled Transition Systems* (LTS) $S = (\mathbf{P}, A, \longrightarrow)$, where $\mathbf{P}$ is a set of process states, A is a set of actions and $\longrightarrow \subseteq \mathbf{P} \times A \times \mathbf{P}$ is a transition relation on processes. We follow the standard practice and write $p \xrightarrow{a} q$ instead of $(p,a,q) \in \longrightarrow$. Because of the covariant-contravariant view, we assume that A is partitioned into A^l and A^r, expressed as $A = A^l \uplus A^r$. As we have already mentioned in the introduction, we will delay the consideration of the general case where we have also bivariant actions in a third class A^{bi} until Section 5.

Covariant-contravariant simulation can now be defined as follows:

Definition 1 *Let $S = (\mathbf{P}, A^l \uplus A^r, \longrightarrow)$ be an LTS. A* covariant-contravariant simulation *over S is a relation $R \subseteq \mathbf{P} \times \mathbf{P}$ such that, whenever $p, q \in \mathbf{P}$ and $p\ R\ q$, we have:*

- *For all $a \in A^r$ and all $p \xrightarrow{a} p'$, there exists some $q \xrightarrow{a} q'$ with $p'\ R\ q'$.*
- *For all $a \in A^l$ and all $q \xrightarrow{a} q'$, there exists some $p \xrightarrow{a} p'$ with $p'\ R\ q'$.*

We will write $p \lesssim_{cc} q$ if there exists a covariant-contravariant simulation R such that $p\ R\ q$.

Remark 1 Note that we call the actions in A^r like that, because for those there is a "plain simulation" from left to right; whereas for the actions in A^l there is an "anti-simulation" from right to left.

It is well known that the relation $\lesssim_{cc}$ is a preorder.

In this study we will be mainly concerned with "finite" properties of systems, which will be either captured by (finite) logic formulae, or by finite processes that can be described by means of process terms.

Definition 2 *Assume that $A = A^l \uplus A^r$. Then the collection of* process terms*, ranged over by p, q etc. is given by the following syntax:*

$$p ::= 0 \mid \omega \mid a.p \mid p + p,$$

where $a \in A$. We denote the set of process terms by $\mathscr{P}$.

The size of a process term is its length in symbols.

We note that our set $\mathscr{P}$ of process terms is basically the set of *BCCSP* terms introduced in [8]. The only addition to the signature of BCCSP is the constant ω, which will be used to denote the least LTS modulo $\lesssim_{cc}$. However, we assume a classification of the actions in two (disjoint) sets, although this is not reflected in the syntactic structure of the terms. Even if $\mathscr{P}$ only contains finite terms, by means of ω we will obtain the full contravariant process which can execute any action at any time.

In [5, 6, 7] we used a more general definition for covariant-contravariant simulations which includes also bivariant actions, but since in the presence of these bivariant actions some technical problems appear (in particular the process ω will not be the least process with respect to the covariant-contravariant simulation preorder), we have preferred to first develop all the results without bivariant actions and, in Section 5, we will describe how they can be extended to a setting with bivariant actions.

Definition 3 *The* operational semantics *of $\mathscr{P}$ is defined by the following rules:*

- $\omega \xrightarrow{b} \omega$ *for all* $b \in A^l$,
- $a.p \xrightarrow{a} p$ *for all* $a \in A$,
- $p \xrightarrow{a} p'$ *implies* $p + q \xrightarrow{a} p'$,
- $q \xrightarrow{a} q'$ *implies* $p + q \xrightarrow{a} q'$.

Observe that if $p \neq \omega$ and $p \xrightarrow{a} p'$, then the size of p' is smaller than the size of p.

It is clear that ω is the least possible element with respect to the cc-simulation preorder. That is, we have $\omega \lesssim_{cc} p$ for any p.

In what follows we assume that A is finite.

3 The covariant-contravariant modal logic

Covariant-contravariant modal logic has been introduced and studied in [7].

Definition 4 Covariant-contravariant modal logic $\mathscr{L}$ *has the following syntax:*

$$\varphi ::= \bot \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [b]\varphi \mid \langle a \rangle \varphi \qquad (a \in A^r, b \in A^l).$$

The operators $\bot$, $\top$, $\wedge$ and $\vee$ have the standard meaning whereas the semantics for the modal operators is defined as follows:

$p \models [b]\varphi$ *if* $p' \models \varphi$ *for all* $p \xrightarrow{b} p'$,

$p \models \langle a \rangle \varphi$ *if* $p' \models \varphi$ *for some* $p \xrightarrow{a} p'$.

We say that a formula φ *is consistent if there is some* p *such that* $p \models \varphi$.

The modal depth *of a formula is the maximum nesting of modal operators in it.*

The covariant-contravariant logic characterizes the covariant-contravariant simulation semantics over image-finite processes. Before we state this result formally we introduce some notation. We define the set of formulae that a process p satisfies by $\mathscr{L}(p) = \{\phi \mid p \models \phi\}$ and the logical preorder $\sqsubseteq_{\mathscr{L}}$ as follows: $p \sqsubseteq_{\mathscr{L}} q$ iff $\mathscr{L}(p) \subseteq \mathscr{L}(q)$. Recall that an LTS is *image finite* iff the set $\{p' \mid p \xrightarrow{a} p'\}$ is finite for each process p and action a.

Now we have the following theorem:

Theorem 1 ([7]) *If the LTS* S *is image finite then* $\lesssim_{cc} = \sqsubseteq_{\mathscr{L}}$ *over* S.

Clearly the processes in $\mathscr{P}$ are image finite.

4 Graphical representation of formulae

Whenever we have a (modal) logic characterizing some semantics for processes, we could look for a single formula that characterizes completely the behaviour of a process logically; this is a so-called *characteristic formula.* This subject has been studied by many authors in the literature, but we will just refer here to the book [2] for more details and further references to the original literature.

It is clear that, since we only allow for finite formulae without any fixed-point operator, we can only treat "finite" processes, such as those definable by our simple process algebra $\mathscr{P}$. However, the recursive definition of the characteristic formulae in what follows gives us immediately the framework for extending our results to finite-state processes following standard lines.

Definition 5 *A formula* $\phi \in \mathscr{L}$ *is a* characteristic formula *for a process* p *iff* $p \models \phi$ *and* $\forall q.(q \models \phi \Rightarrow p \lesssim_{cc} q)$.

In what follows, we write $\phi \leq \psi$ if $\{p \in P \mid p \models \phi\} \subseteq \{p \in P \mid p \models \psi\}$. We say that ϕ and ψ are logically equivalent, written $\phi \equiv \psi$, iff $\phi \leq \psi$ and $\psi \leq \phi$.

Lemma 1 *The following statements hold.*

1. *A formula* $\phi \in \mathscr{L}$ *is a characteristic formula for a process* p *iff* $\forall q.(q \models \phi \Leftrightarrow p \lesssim_{cc} q)$.
2. *Assume that* $\chi(p)$ *and* $\chi(q)$ *are characteristic formulae for processes* p *and* q*, respectively. Then, we have that*
$$p \lesssim_{cc} q \text{ iff } \chi(q) \leq \chi(p).$$
3. *A characteristic formula for a process* p *is unique up to logical equivalence.*

Proof.

1. First assume that ϕ is a characteristic formula for a process p. By definition $\forall q.(q \models \phi \Rightarrow p \lesssim_{cc} q)$ holds. We have to prove that $\forall q.(p \lesssim_{cc} q \Rightarrow q \models \phi)$. To this end, assume that $p \lesssim_{cc} q$. As $p \models \phi$, by Theorem 1 we have that $q \models \phi$ and we are done.

   For the converse, as $p \lesssim_{cc} p$ we have that $p \models \phi$ and the result follows.

2. Assume that $\chi(p)$ and $\chi(q)$ are characteristic formulae for processes p and q, respectively. First assume that $p \lesssim_{cc} q$ and that $r \models \chi(q)$. By Definition 5, $q \lesssim_{cc} r$ and thus $p \lesssim_{cc} r$. By the previous clause of the Lemma, also $r \models \chi(p)$. As r was arbitrary, this shows that $\chi(q) \leq \chi(p)$. Next, assume that $\chi(q) \leq \chi(p)$. As $q \models \chi(q)$ then $q \models \chi(p)$, and by definition of the characteristic formula, $p \lesssim_{cc} q$.
3. This claim follows directly from statement 2 above. □

As a characteristic formula for a process p is unique up to logical equivalence, we can denote it by $\chi(p)$ unambiguously. The next lemma tells us that $\chi(p)$ exists for each process $p \in \mathscr{P}$.

Lemma 2 *The characteristic formula for a process $p \in \mathscr{P}$ can be obtained recursively as*

$$\begin{aligned} \chi(p) &= \bigwedge_{p \xrightarrow{a} p', a \in A^r} \langle a \rangle \chi(p') \wedge \bigwedge_{b \in A^l} [b]\Big(\bigvee_{p \xrightarrow{b} p'} \chi(p') \Big), \text{ if } p \neq \omega. \\ \chi(\omega) &= \top. \end{aligned}$$

Proof. First we prove that $p \models \chi(p)$, for each p. This follows by a simple induction on the size of p.

Next we prove that, for any q, $q \models \chi(p)$ implies $p \lesssim_{cc} q$ by induction on the size of q.

First we note that if $p = \omega$ then $\chi(\omega) = \top$ and $\omega \lesssim_{cc} q$; hence we obtain the result. Also, for the case $p = 0$, we have that $\chi(0)$ is equivalent to $\bigwedge_{b \in A^l} [b]\bot$. Thus if $q \models \chi(0)$, then the process q cannot perform any $b \in A^l$. This yields that $0 \lesssim_{cc} q$.

Now, let p be a process different from 0 and ω, and assume that $q \models \chi(p)$. First suppose that $p \xrightarrow{a} p'$ for some p' and some $a \in A^r$. As $q \models \bigwedge_{p \xrightarrow{a} p', a \in A^r} \langle a \rangle \chi(p')$, this implies that there is some $q \xrightarrow{a} q'$ with $q' \models \chi(p')$. Then, by induction, $p' \lesssim_{cc} q'$.

Next, assume that $q \xrightarrow{b} q'$, for some q' and $b \in A^l$. As $q \models \bigwedge_{b \in A^l} [b](\bigvee_{p \xrightarrow{b} p'} \chi(p'))$, we can conclude that $q' \models \chi(p')$, for some p' with $p \xrightarrow{b} p'$. Again, by induction, we conclude $p' \lesssim_{cc} q'$. □

Next we consider the converse problem, we want to represent a formula by a process, or at least by a finite set of processes.

Definition 6 *A formula ϕ is* represented by a (single) process *p if*

$$\forall q \in \mathscr{P}.\ [q \models \phi \text{ iff } p \lesssim_{cc} q].$$

A formula ϕ is represented by a finite set *$M \subseteq \mathscr{P}$ of processes if*

$$\forall q \in \mathscr{P}.\ [q \models \phi \text{ iff } \exists p \in M.\ p \lesssim_{cc} q].$$

It is clear that p represents ϕ iff $\{p\}$ represents ϕ. Moreover, the empty set of processes represents the formula $\bot$.

The following lemma connects the notion of "graphical representation" of formulae with that of characteristic formula for processes.

Lemma 3 *We have the following properties:*

1. *p represents ϕ iff $\phi \equiv \chi(p)$.*
2. *If $M \subseteq \mathscr{P}$ is finite and ϕ is a formula then*

$$M \text{ represents } \phi \text{ iff } \phi \equiv \bigvee_{p \in M} \chi(p).$$

Proof.

1. It follows directly from the definitions of these two concepts and Lemma 1.
2. For any $q \in \mathscr{P}$ we proceed as follows:

$$\exists p \in M.p \lesssim_{cc} q \Leftrightarrow \exists p \in M.q \models \chi(p) \Leftrightarrow q \models \bigvee_{p \in M} \chi(p).$$

Now the statement of the lemma follows easily from this fact and Definition 6. □

We want to characterize the set of formulae that can be represented by a finite set of processes, and in particular by a single process. For this purpose we introduce some notions of normal form for logical formulae.

Definition 7 *1. A formula ϕ is in normal form if it has the form*

$$\phi = \bigvee_{i \in I} (\bigwedge_{j \in J_i} \langle a^i_j \rangle \phi^i_j \wedge \bigwedge_{k \in K_i} [b^i_k] \psi^i_k).$$

where all ϕ^i_j and ψ^i_k are also in normal form. In particular, $\bot$ is obtained when $I = \emptyset$ and $\top$ when $I = \{1\}$ and $J_1 = K_1 = \emptyset$.

2. A formula ψ is in strong normal form if it has the form

$$\psi = \bigvee_{i \in I} \phi_i \ ,$$

where each ϕ_i is in unary strong normal form. A formula ϕ is in unary strong normal form if it is $\top$ or it has the form

$$\phi = \bigwedge_{j \in J} \langle a_j \rangle \phi_j \wedge \bigwedge_{b \in A^I} [b] \psi_b,$$

where every ϕ_j is in unary strong normal form and every ψ_b is in strong normal form.

We note that any unary strong normal form different from $\top$ can equivalently be written as

$$\phi = \bigwedge_{j \in J} \langle a_j \rangle \phi_j \wedge \bigwedge_{b \in A^I} [b] \bigvee_{k \in K_b} \psi^k_b,$$

where every ϕ_j and every ψ^k_b are in unary strong normal form, thus avoiding the introduction of strong normal forms.

Remark 2 It is not hard to see that each unary strong normal form is consistent. See also Theorem 2 to follow.

Clearly the characteristic formulae of processes are in unary strong normal form. Therefore, by Lemma 3, it is a necessary condition for a formula to be representable by a single process that it has an equivalent unary strong normal form. We will show that this is also a sufficient condition for this to hold for any consistent formula.

Theorem 2 *A unary strong normal form*

$$\phi = \bigwedge_{j\in J} \langle a_j\rangle \phi_j \wedge \bigwedge_{b\in A^l} [b] \bigvee_{k\in K_b} \psi_b^k$$

is represented by the process defined recursively by

$$\begin{aligned} \theta(\phi) &= \sum_{j\in J} a_j.\theta(\phi_j) + \sum_{b\in A^l}\sum_{k\in K_b} b.\theta(\psi_b^k), \quad \text{if } \phi \neq \top \\ \theta(\top) &= \omega. \end{aligned}$$

In particular ϕ is the characteristic formula for $\theta(\phi)$ (up to logical equivalence). Note that even if in the formal expression above there is a summand for each $b \in A^l$, only those b's such that $K_b \neq \emptyset$ will finally appear as summands of $\theta(\phi)$.

Proof. First we prove that $\theta(\phi) \models \phi$ by induction on the modal depth of ϕ. If $\phi = \top$ we have that obviously $\theta(\phi) = \omega \models \phi = \top$. For the inductive step first we note that $\theta(\phi) \xrightarrow{a_j} \theta(\phi_j)$ for all $j \in J$. By induction, $\theta(\phi_i) \models \phi_i$. Next assume that $\theta(\phi) \xrightarrow{b} p$ for some $b \in A^l$ and some p. We have that $p = \theta(\psi_b^k)$ for some $k \in K_b$. By induction $\theta(\psi_b^k) \models \psi_b^k$ and therefore $\theta(\psi_b^k) \models \bigvee_{k\in K_b} \psi_b^k$.

Next we prove that if $q \models \phi$ then $\theta(\phi) \lesssim_{cc} q$. Towards proving this claim, assume that $q \models \phi$. Again we proceed by induction on the modal depth of ϕ.

First assume that $\theta(\phi) \xrightarrow{a} p'$ for some $a \in A^r$ and process term p'. Then $a = a_j$ for some $j \in J$ and $p' = \theta(\phi_j)$. As $q \models \phi$, we have that $q \xrightarrow{a_j} q'$ for some q' with $q' \models \phi_j$. By induction, $\theta(\phi_j) \lesssim_{cc} q'$, as required.

Now assume that $q \xrightarrow{b} q'$ for some $b \in A^l$. As $q \models \phi$ we have that $q' \models \psi_b^k$ for some $k \in K$. Now $\theta(\phi) \xrightarrow{b} \theta(\psi_b^k)$ and, by the induction hypothesis, we have $\theta(\psi_b^k) \lesssim_{cc} q'$, as required.

This proves that ϕ is the characteristic formula for $\theta(\phi)$ and therefore, by Lemma 3, that $\theta(\phi)$ represents ϕ. □

Next, we will show that any formula has an equivalent strong normal form and therefore can always be represented by a (possibly empty) finite set of processes. To derive this result we will use several standard equivalences between formulae.

Lemma 4 *The following statements hold.*

1. *$\wedge$ and $\vee$ are associative, commutative and idempotent.*
2. *$\wedge$ distributes over $\vee$, and $\vee$ distributes over $\wedge$.*
3. *$\phi \vee \top \equiv \top$, $\phi \vee \bot \equiv \phi$, $\phi \wedge \top \equiv \phi$, and $\phi \wedge \bot \equiv \bot$.*
4. *$[b]\top \equiv \top$.*
5. *$[b]\phi \wedge [b]\psi \equiv [b](\phi \wedge \psi)$ for $b \in A^l$.*
6. *$\langle a\rangle\phi \vee \langle a\rangle\psi \equiv \langle a\rangle(\phi \vee \psi)$ for $a \in A^r$.*

Proof. The first three collections of equalities are straightforward and well known, so we omit their proofs.

- $[b]\top \equiv \top$. We have $p \models [b]\top$ iff $p' \models \top$ for all $p \xrightarrow{b} p'$. Therefore, the condition is satisfied whenever $p \xrightarrow{b} p'$, and it is vacuously true when $p \overset{b}{\not\longrightarrow}$.

- $[b]\phi \wedge [b]\psi \equiv [b](\phi \wedge \psi)$. We have $p \models ([b]\phi \wedge [b]\psi)$ iff $p' \models \phi$ for all $p \xrightarrow{b} p'$ and $p' \models \psi$ for all $p \xrightarrow{b} p'$, iff $p' \models (\phi \wedge \psi)$ for all $p \xrightarrow{b} p'$, iff $p \models [b](\phi \wedge \psi)$.
- $\langle a\rangle\phi \vee \langle a\rangle\psi \equiv \langle a\rangle(\phi \vee \psi)$. We have $p \models \langle a\rangle\phi \vee \langle a\rangle\psi$ iff there exists $p \xrightarrow{a} p'$ such that $p' \models \phi$ or there exists $p \xrightarrow{a} p''$ such that $p'' \models \psi$, that is, iff there exists some $p \xrightarrow{a} p'_0$ such that $p'_0 \models \phi$ or $p'_0 \models \psi$. This holds iff $p \models \langle a\rangle(\phi \vee \psi)$. □

Lemma 5 *Every formula ϕ has an equivalent strong normal form with no larger modal depth.*

Proof. First we prove by induction on the modal depth, using 1-3 of Lemma 4, that ϕ has an equivalent normal form with the same modal depth. To prove the main statement we can therefore assume that ϕ is in normal form. We proceed by induction on the modal depth $md(\phi)$. The base case $md(\phi) = 0$ ($\phi \equiv \bot$ and $\phi \equiv \top$) follows immediately.

Next let us assume that

$$\phi = \bigvee_{i\in I}\Big(\bigwedge_{j\in J_i}\langle a^i_j\rangle\phi^i_j \wedge \bigwedge_{k\in K_i}[b^i_k]\psi^i_k\Big).$$

By Lemma 4, using 4 and 5 and the standard laws described in 1-3, ϕ can be rewritten into an equivalent formula of the form

$$\phi = \bigvee_{i\in I}\Big(\bigwedge_{j\in J_i}\langle a^i_j\rangle\phi^i_j \wedge \bigwedge_{b\in A^l}[b]\psi^i_b\Big)$$

where $md(\psi^i_b) \leq \sup\{md(\psi^i_k) \mid k \in K_i\}$ (we note that some of the $[b]\psi^i_b$s may have the form $[b]\top$, which is equivalent to $\top$). Therefore, by the induction hypothesis, we may assume that ϕ^i_j and ψ^i_b are in strong normal form. Next we use Lemma 4.6 to remove all the occurrences of $\vee$ that are guarded by $\langle a\rangle$, for some $a \in A^r$ in each $\bigwedge_{j\in J_i}\langle a^i_j\rangle\phi^i_j$. The result for each i is of the form $\bigwedge_{j\in J_i}(\bigvee_{l\in L_j}\langle a^i_j\rangle\phi^{l,i}_j)$, where each $\phi^{l,i}_j$ is in a unary strong normal form. By repeated use of distributivity, the whole formula can be rewritten as

$$\phi = \bigvee_{r\in R}\Big(\bigwedge_{s\in S_r}\langle a^r_s\rangle\alpha^r_s \wedge \bigwedge_{b\in A^l}[b]\bigvee_{t\in T^r_b}\beta^r_{b,t}\Big)$$

where each α^s_r and $\beta^{b,t}_r$ is a unary strong normal form. Finally we note that the operations described above do not increase the modal depth. □

Now we will relate our result to the one in Boudol and Larsen's paper [4].

Definition 8 *A formula ϕ is* prime *if the following holds:*

$$\forall \phi_1, \phi_2 \in \mathscr{L}.\ \phi \leq \phi_1 \vee \phi_2 \text{ implies } \phi \leq \phi_1 \text{ or } \phi \leq \phi_2.$$

Theorem 3 *A formula ϕ can always be represented by a finite set of processes. It can be represented by a single process if and only if it is consistent and prime.*

Proof. By Lemma 5, $\phi \equiv \phi_1 \vee \ldots \vee \phi_n$ where each ϕ_i, $1 \leq i \leq n$, is in unary strong normal form. By Theorem 2, $\phi_i \equiv \chi(p_i)$ for some p_i for each $1 \leq i \leq n$, and therefore $\phi \equiv \chi(p_1) \vee \ldots \vee \chi(p_n)$. The first statement now follows from Lemma 3.2.

Towards proving the second statement, first assume that $\phi \equiv \chi(p_1) \vee \ldots \vee \chi(p_n)$ is prime. This implies that $\phi \leq \chi(p_i) \leq \phi$, for some $i \in \{1, \ldots, n\}$, which in turn implies that $\phi \equiv \chi(p_i)$.

Next assume that ϕ is represented by some process p or equivalently that $\phi \equiv \chi(p)$. Now assume that $\chi(p) \leq \phi_1 \vee \phi_2$. As $p \models \chi(p)$, this implies that $p \models \phi_1 \vee \phi_2$ or equivalently that either $p \models \phi_1$ or $p \models \phi_2$. Without loss of generality, we can assume that $p \models \phi_1$. Now assume that $r \models \chi(p)$. Then $p \lesssim_{cc} r$ and by Theorem 1 this implies that $r \models \phi_1$. Since r was arbitrary, this proves that $\phi \equiv \chi(p) \leq \phi_1$. Hence ϕ is prime, which was to be shown. □

5 Considering bivariant actions

Originally [5, 6, 7], the theory of covariant-contravariant semantics also considered bivariant actions in A^{bi}, so that we had a partition of A into $\{A^r, A^l, A^{bi}\}$ (called the signature of the LTS), and the definition of covariant-contravariant simulations imposed the following two conditions:

- For all $a \in A^r \cup A^{bi}$ and all $p \xrightarrow{a} p'$, there exists some $q \xrightarrow{a} q'$ with $p' \, R \, q'$.
- For all $a \in A^l \cup A^{bi}$ and all $q \xrightarrow{a} q'$, there exists some $p \xrightarrow{a} p'$ with $p' \, R \, q'$.

When we have in our signature bivariant actions we cannot get directly the graphical representation results that we have presented in Section 4. This is so because bivariant actions cannot be under approximated, as a consequence of the well known result that bisimilarity is an equivalence relation and not a plain preorder. In order to maintain our results we mandatorily need that notion of approximation. We obtain it by decomposing each bivariant action a into a pair of actions, one covariant, a^r, and another contravariant, a^l. Technically, we define an embedding of the set of processes over an arbitrary signature $A = \{A^r, A^l, A^{bi}\}$ into that corresponding to a new signature $\bar{A} = \{\bar{A^r}, \bar{A^l}, \emptyset\}$. The latter does not include any bivariant action, and then we can apply to it our graphical representation results, that then can be transfered to the original signature by means of the defined embedding.

In [1] we presented transformations from LTSs to Modal Transition Systems (MTSs), and vice versa, named $\mathscr{M}$ and $\mathscr{C}$, respectively. We proved that both preserve and reflect the covariant-contravariant logic and simulation preorder. Applying these two transformations in a row we did not obtain the identity function, but instead a transformation $\mathscr{T}_0 = \mathscr{C} \circ \mathscr{M}$ that transforms an LTS with bivariant actions into another LTS without them. Since composition preserves the good properties of $\mathscr{C}$ and $\mathscr{M}$, $\mathscr{T}_0$ also has these properties.

Next we give a direct definition of $\mathscr{T}_0$.

Definition 9 *Let T be an LTS with the signature $A = \{A^r, A^l, A^{bi}\}$. The LTS $\mathscr{T}_0(T)$ with signature $\hat{A} = \{\hat{A^r}, \hat{A^l}, \emptyset\}$, where $\hat{A^r} = \{d^r \mid d \in A^r \cup A^{bi}\}$ and $\hat{A^l} = \{d^l \mid d \in A^r \cup A^l \cup A^{bi}\}$, is constructed as follows:*

- *The set of states of $\mathscr{T}_0(T)$ is the same as the one of T plus a new state u.*
- *For each transition $p \xrightarrow{d} p'$ in T , add a transition $p \xrightarrow{d^l} p'$ in $\mathscr{T}_0(T)$.*
- *For each transition $p \xrightarrow{d} p'$ in T with $d \in A^r \cup A^{bi}$, add a transition $p \xrightarrow{d^r} p'$ in $\mathscr{T}_0(T)$.*
- *For each $a \in A^r$ and state p, add the transition $p \xrightarrow{a^l} u$ to $\mathscr{T}_0(T)$, as well as transitions $u \xrightarrow{d^l} u$, for each action $d \in A$.*

Note that each $c \in A^{bi}$ is "encoded" by means of a pair of new actions (c^r, c^l). Moreover, as a consequence of the general definition of $\mathscr{M}$, for each $a \in A^r$, together with a^r, which is its "natural" encoding an additional $a^l \in A^l$, coupled with it, is introduced. Finally, the behaviour of the "extra" state u is defined by ω.

Based on this transformation, we have designed a direct encoding of LTSs over a signature $A = \{A^r, A^l, A^{bi}\}$ by means of LTSs over an adequate signature $\bar{A} = \{\bar{A^r}, \bar{A^l}, \emptyset\}$. As above, for each $c \in A^{bi}$ in the original signature, we introduce a pair of (new) actions, as the following definition makes precise.

Definition 10 *Let T be an LTS with signature $A = \{A^r, A^l, A^{bi}\}$. The LTS $\mathscr{T}(T)$, with signature $\bar{A} = \{\bar{A^r}, \bar{A^l}, \emptyset\}$, where $\bar{A^r} = A^r \cup \{c^r \mid c \in A^{bi}\}$ and $\bar{A^l} = A^l \cup \{c^l \mid c \in A^{bi}\}$, is constructed as follows:*

- *The set of states of $\mathscr{T}(T)$ is the same as that of T.*

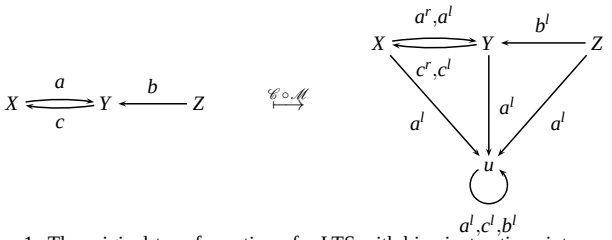


Figure 1: The original transformation of a LTS with bivariant actions into another without them, assuming $A^r = \{a\}$, $A^l = \{b\}$ and $A^{bi} = \{c\}$.

- *All the transitions from T with label in $A^r \cup A^l$ are in $\mathscr{T}(T)$.*
- *For each transition $p \xrightarrow{c} p'$ in T with $c \in A^{bi}$, we add $p \xrightarrow{c^r} p'$ and $p \xrightarrow{c^l} p'$ to $\mathscr{T}(T)$.*

The transformation above produces an LTS without bivariant actions more closely related to the original covariant-contravariant LTS than that produced by $\mathscr{T}_0$ (compare Figure 2 with Figure 1). Note that the class of LTSs with signature $\overline{A}$ that satisfy that $p \xrightarrow{c^r} p'$ if and only if $p \xrightarrow{c^l} p'$, for all $p, p' \in \mathbf{P}$, and all $c \in A^{bi}$; is exactly the class of processes that are the representation of some LTS with signature A.

To translate modal formulae we have just to adopt the right modality for each action, as the following definition makes precise.

Definition 11 *Let us extend $\mathscr{T}$ to translate modal formulae over the modal logic for LTS over A into modal formulae over the modal logic for LTS over $\overline{A}$, as follows:*

- $\mathscr{T}(\bot) = \bot$.
- $\mathscr{T}(\top) = \top$.
- $\mathscr{T}(\varphi \wedge \psi) = \mathscr{T}(\varphi) \wedge \mathscr{T}(\psi)$.
- $\mathscr{T}(\varphi \vee \psi) = \mathscr{T}(\varphi) \vee \mathscr{T}(\psi)$.
- $\mathscr{T}(\langle a \rangle \varphi) = \langle a \rangle \mathscr{T}(\varphi)$, *if* $a \in A^r$.
- $\mathscr{T}(\langle c \rangle \varphi) = \langle c^r \rangle \mathscr{T}(\varphi)$, *if* $c \in A^{bi}$.
- $\mathscr{T}([b]\varphi) = [b]\mathscr{T}(\varphi)$, *if* $b \in A^l$.
- $\mathscr{T}([c]\varphi) = [c^l]\mathscr{T}(\varphi)$, *if* $c \in A^{bi}$.

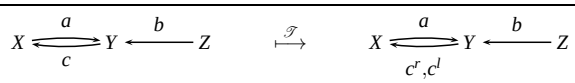


Figure 2: The new transformation $\mathscr{T}(T)$ of an LTS with bivariant actions into another without them, assuming $A^r = \{a\}$, $A^l = \{b\}$ and $A^{bi} = \{c\}$.

In order to show that $\mathscr{T}$ preserves and reflects the cc-simulation preorder, we compare $\mathscr{T}(T)$ with $\mathscr{T}_0(T)$ and we prove a more general result.

Definition 12 *Given a signature $\{A^r, A^l, \emptyset\}$ and $c^l \in A^l$ we define the transformation $\mathscr{T}^+_{c^l}$ as that which given an LTS T with that signature adds a new state u whose behaviour is that defined by ω, and a new transition labelled by c^l from each state of T to u.*

Proposition 1 *$\mathscr{T}^+_{c^l}$ preserves and reflects the cc-simulation preorder when applied to a system that does not contain any c^l transition.*

Proof. We will see that R is a cc-simulation in T if and only if $R \cup \{(u,u)\}$ is a cc-simulation in $\mathscr{T}^+_{c^l}(T)$. The result is immediate by simply observing that for a-transitions, with $a \neq c^l$, the leaving of any state p with $p \neq u$ are exactly the same in T and $\mathscr{T}^+_{c^l}(T)$, while for any such state we always have $p \xrightarrow{c^l} u$ in $\mathscr{T}^+_{c^l}(T)$. □

Corollary 1 *Let T be an LTS with signature $\{A^r, A^l, A^{bi}\}$. Then, for any two states p and q of T, we have $p \lesssim_{cc} q$ in $\mathscr{T}(T)$ if and only if $p \lesssim_{cc} q$ in $\mathscr{T}_0(T)$.*

Proof. Note that $\mathscr{T}(T)$ is a $\{\overline{A}^r, \overline{A}^l, \emptyset\}$-LTS, while $\mathscr{T}_0(T)$ is an $\{\hat{A}^r, \hat{A}^l, \emptyset\}$-LTS, where $\hat{A}^r = \{a^r \mid a \in A^r \cup A^{bi}\}$ and $\hat{A}^l = \overline{A}^l \cup \{a^l \mid a \in A^r\}$. This means that we can also see $\mathscr{T}(T)$ as an $\{\hat{A}^r, \hat{A}^l, \emptyset\}$-LTS if we rename each $a \in A^r$ into the corresponding $a^r \in \hat{A}^r$. Then, we can apply $\mathscr{T}^+_{a^l}$ for each $a \in A^r$ in a row, thus getting a transformed system $\mathscr{T}^+(T)$. All along these applications we are under the hypothesis of Proposition 1. Moreover, the only differences between $\mathscr{T}^+(T)$ and $\mathscr{T}_0(T)$ are the collection of a^l-transitions paired with the a^r-transitions in T, with $a \in A^r$. But since for any state p of $\mathscr{T}^+(T)$ we have $p \xrightarrow{a^l} u$, for all $a^l \in \{a^l \mid a^r \in A^r\}$, we immediately conclude that the identity is a cc-simulation in both directions (up-to the indicating renaming) between the states of $\mathscr{T}^+(T)$ and those in $\mathscr{T}_0(T)$, from which we finally obtain that $p \lesssim_{cc} q$ in $\mathscr{T}(T)$ iff $p \lesssim_{cc} q$ in $\mathscr{T}_0(T)$. □

Corollary 2 *Our transformation $\mathscr{T}$ preserves and reflects the cc-simulation preorder, that is, for each LTS T and for all states p and q in T, it holds that $p \lesssim_{cc} q$ in T if, and only, if $p \lesssim_{cc} q$ in $\mathscr{T}(T)$.*

Proof. We just need to combine Proposition 1 and Corollary 1. □

Proposition 2 *$\mathscr{T}$ preserves and reflects the cc-logic, that is, for each LTS T, any state p and all covariant-contravariant formula φ in T, it holds that $p \models \varphi$ in T if, and only if, $p \models \mathscr{T}(\varphi)$ in $\mathscr{T}(T)$.*

Proof. We proved in [1] the corresponding result for $\mathscr{T}_0$ and the transformation $\mathscr{T}_0$ which is defined on logic formulae exactly as $\mathscr{T}$, but renaming again each $a \in A^r$ into a^r. From the definitions of $\mathscr{T}$ and $\mathscr{T}_0$ we immediately conclude that a^l-transitions with $a \in A^r$ do not play any role in the satisfaction of any formula $\mathscr{T}(\varphi)$, and then the result follows from that proved in [1]. □

After the representation of a bivariant action $c \in A^{bi}$ as a pair (c^r, c^l) with $c^r \in \overline{A}^r$ and $c^l \in \overline{A}^l$, we have that c^l under-approximates c, whereas c^r over-approximates c. This means in particular that we have $c^l 0 \lesssim_{cc} c^l 0 + c^r 0 \lesssim_{cc} c^r 0$ and, more generally, $c^l p \lesssim_{cc} c^l p + c^r q \lesssim_{cc} c^r q$, for all processes p and q. Therefore, once we have separated the covariant and contravariant characters of bivariant actions we achieve a greater flexibility which allows us to consider "non-balanced" processes where these two characters do not go always together, thus producing over and under-approximations when needed.

Discussion It is interesting to compare our new transformation $\mathscr{T}$ with the original transformation $\mathscr{T}_0$ from [1]. The first aims to obtain a representation over the signature $\{\overline{A}^r, \overline{A}^l, \emptyset\}$ that is as simple as

possible, and this is why we do not introduce a^l when $a \in A^r$. Instead, we can see the result of the transformation $\mathscr{T}_0$ as a process in the "uniform" signature $\tilde{A} = \{\tilde{A}^r, \tilde{A}^l, \emptyset\}$, with $\tilde{A}^r = \{a^r \mid a \in A^r \cup A^l \cup A^{bi}\}$ and $\tilde{A}^l = \{a^l \mid a \in A^r \cup A^l \cup A^{bi}\}$. It is true that the actions b^r with $b \in A^l$ do not appear in $\mathscr{T}_0(T)$, but even so we can consider any $\mathscr{T}_0(T)$ as a process for $\tilde{A}$. Obviously, this is also the case for $\mathscr{T}(T)$, where the actions a^l with $a \in A^r$ do not appear either. Both $\mathscr{T}_0(T)$ and $\mathscr{T}(T)$ were "good" representations of T, as stated above, however it is clear that we do not have $\mathscr{T}_0(T) \equiv_{cc} \mathscr{T}(T)$. Instead, $\mathscr{T}_0(T) \lesssim_{cc} \mathscr{T}(T)$, and in fact $\mathscr{T}_0(T)$ is the least process with respect to $\lesssim_{cc}$, for the uniform signature $\tilde{A}$ that has the good properties stated in the paper. Note that, instead, b^r-transitions for $b \in A^l$ do not need to be introduced at all, since any addition of a covariant transitions produces a $\lesssim_{cc}$-greater process.

Therefore, the original transformation $\mathscr{T}_0$ would be indeed the adequate one if we wanted to obtain an embedding of the class of processes for any signature into that corresponding to the uniform signature $\tilde{A}$ defined above, where all the actions can be interpreted as the covariant and contravariant parts of the actions in a set A.

To conclude the section we explore the set of systems for any signature $\bar{A} = \{\bar{A}^r, \bar{A}^l, \emptyset\}$. Some of them, but not all, are equivalent to the representation of a system for the original alphabet A. Whenever that is not the case we would need to remove (or add) some transitions labelled by the created actions in $\{c^r, c^l \mid c \in A^{bi}\}$ in order to obtain a system that is equivalent to the representation of some process. In the following proposition we give an algorithm for obtaining a system for the original signature A to which a given system for the signature $\bar{A}$ is equivalent, whenever such a system exists. To make possible a proof by (structural) induction, we will only present the result for process terms in $\mathscr{P}$.

Proposition 3 *Let $A = \{A^r, A^l, A^{bi}\}$ be a signature and $\bar{A} = \{\bar{A}^r, \bar{A}^l, \emptyset\}$ be the associated signature without bivariant actions. Let $p, q \in \mathscr{P}$ be process terms for $\bar{A}$ such that q is the representation of some process for the signature A. Let us assume that $p \equiv_{cc} q$. Then it is possible to transform p into the representation p_{bi} of some process term for A, simply by adding or removing some transitions labelled by actions in $\{c^r, c^l \mid c \in A^{bi}\}$.*

Proof. The proof is done by structural induction.

- If $p = 0$ or $p = \omega$ we can take $p_{bi} = p$.
- In the general case, we exploit the fact that whenever $a \in \bar{A}^r$, if $q' \lesssim_{cc} p'$ then $ap' + aq' \equiv_{cc} ap'$ (and dually, when $b \in \bar{A}^l$, $bp' + bq' \equiv_{cc} bq'$). This means that from any term for $\bar{A}$ we can remove all the summands aq'' (resp. bp'') such that ap'' is not a maximal a-summand of p' with respect to $\lesssim_{cc}$ (resp. bp'' is not a minimal a-summand), obtaining a $\equiv_{cc}$-equivalent process. So, we start by removing all the non-maximal a-summands with $a \in A^r$, and all the non-minimal b-summands with $b \in \bar{A}^l$ of any subterm of p. By abuse of notation, we will still denote the obtained process by p, and we still have $p \equiv_{cc} q$.

  Now, for any a-summand of p with $a \in \bar{A}^r$, $p = p' + ap''$, there is some $q \xrightarrow{a} q''$ with $p'' \lesssim_{cc} q''$. But also, since $p \equiv_{cc} q$, starting with $q \xrightarrow{a} q''$ there must exist some $p \xrightarrow{a} p'''$ with $q'' \lesssim_{cc} p'''$, but then $p'' \lesssim_{cc} p'''$, and since p'' was maximal we can assume that $p''' = p''$, and then we also have $p'' \equiv_{cc} q''$. The same is true for all the b-summands with $b \in \bar{A}^l$, and this means that we can apply the induction hypothesis to all the derivatives of p.

  Moreover, for each ap' summand with $a = c^r$ we can add to p the summand $c^l p'$ and we obtain $p \equiv_{cc} p + c^l p'$. Indeed, we have trivially $p + c^l p' \lesssim_{cc} p$, and to prove that $p \lesssim_{cc} p + c^l p'$ we check $q \lesssim_{cc} p + c^l p'$. We only need to see that for any transition $p + c^l p' \xrightarrow{c^l} p'$ there is some $q \xrightarrow{c^l} q'$

with $q' \lesssim_{cc} p'$. We use again the maximality of the summand $c^r p'$ and we obtain, as above, that there is some $c^r q'$ summand of q with $q' \lesssim_{cc} p'$. But since q was the representation of some process for A, it has also a summand $c^l q'$ as required above.

The obtained process has already its c^r and c^l transitions, with $c \in A^{bi}$, paired at its first level, and then we simply need to apply the induction hypothesis to conclude the proof. □

Remark 3 Although the proposition above assumes that the considered process was equivalent to the representation of some process for A, it is easy to use it as a decision algorithm to check that property: we apply the algorithm to the given process p and check if the obtained process p' is $\equiv_{cc}$-equivalent to it, if that is not the case then p is not equivalent to the representation of any process for the signature A.

6 Conclusions and future work

In [1] we studied the relationships between the notion of refinement over modal transition systems, and the notions of covariant-contravariant simulation and partial bisimulation over labelled transition systems. Here we have continued that work by looking for the "graphical" representation of the covariant-contravariant modal formulae by means of terms, as it was done in [3] for the case of modal transition systems. For technical reasons, we had first to restrict ourselves to the case in which we have no bivariant actions. Afterwards, we argued that the general case can, in some sense, be "reduced" to the one we dealt with in Section 4 by defining a semantic-preserving transformation between covariant-contravariant systems with bivariant actions, and covariant-contravariant systems without them.

The idea was to separate each bivariant action into its covariant and its contravariant parts. As a matter of fact, we believe that this idea might be useful not only for obtaining theoretical results, as we have done here, but also for applications. Most of the studies on process algebras and their semantics assume the bivariant behaviour of all the actions. It is true that in some studies (see for example [13]) we have a classification of actions, as we have also done in [1] and in this paper. But now we are proposing to exploit the relationships between the different classes of actions.

As future work, it would be interesting to obtain a direct characterization of the formulae that are graphically representable in a setting with bivariant actions. Such a direct characterization will also pave the way towards a more general theory of "graphical characterizations" of formulae in modal logics of processes, of which the result by Boudol and Larsen and ours are special cases.

Of course, one of the directions in which we plan to continue our studies is that related with the logical characterization of the semantics, and in particular the connections between logical formulae and terms established by characteristic formulae and graphical representations. The combination of these two frameworks is also an interesting challenge. In particular, we plan some extensions of the recent work by Lüttgen and Vogler [11, 12] to the case of covariant-contravariant systems.

References

[1] Luca Aceto, Ignacio Fábregas, David de Frutos Escrig, Anna Ingólfsdóttir & Miguel Palomino (2011): *Relating modal refinements, covariant-contravariant simulations and partial bisimulations*. In *Fundamentals of Software Engineering, FSEN 2011*, LNCS, Springer.To appear.

[2] Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen & Jiří Srba (2007): *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press.

[3] J. Baeten, D. van Beek, B. Luttik, J. Markovski & J. Rooda (2010): *Partial Bisimulation*. SE Report 2010-04, Department of Mechanical Engineering, Eindhoven University of Technology, http://se.wtb.tue.nl/sereports.

[4] Gérard Boudol & Kim Gulstrand Larsen (1992): *Graphical versus logical specifications*. *Theoretical Computer Science* 106(1), pp. 3–20, doi:10.1016/0304-3975(92)90276-L.

[5] Ignacio Fábregas, David de Frutos-Escrig & Miguel Palomino (2009): *Non-strongly Stable Orders Also Define Interesting Simulation Relations*. In *CALCO'09*, *LNCS* 5728, Springer, pp. 221–235, doi:10.1007/978-3-642-03741-2_16.

[6] Ignacio Fábregas, David de Frutos-Escrig & Miguel Palomino (2010): *Equational Characterization of Covariant-Contravariant Simulation and Conformance Simulation Semantics*. In *SOS'10*, *EPTCS* 32, pp. 1–14, doi:10.4204/EPTCS.32.1.

[7] Ignacio Fábregas, David de Frutos-Escrig & Miguel Palomino (2010): *Logics for Contravariant Simulations*. In *FORTE-FMOODS 2010*, *LNCS* 6117, Springer, pp. 224–231, doi:10.1007/978-3-642-13464-7_18.

[8] R. J. van Glabbeek (2001): *The linear time-branching time spectrum I: The semantics of concrete, sequential processes*. In J. A. Bergstra, A. Ponse & S. A. Smolka, editors: *Handbook of process algebra*, North-Holland, pp. 3–99.

[9] Kim Guldstrand Larsen (1989): *Modal Specifications*. In *Automatic Verification Methods for Finite State Systems*, *LNCS* 407, Springer, pp. 232–246, doi:10.1007/3-540-52148-8_19.

[10] Kim Guldstrand Larsen & Bent Thomsen (1988): *A Modal Process Logic*. In: *LICS 1988*, IEEE Computer Society, pp. 203–210, doi:10.1109/LICS.1988.5119.

[11] Gerald Lüttgen & Walter Vogler (2009): *Safe Reasoning with Logic LTS*. In *SOFSEM 2009*, *LNCS* 5404, Springer, pp. 376–387, doi:10.1007/978-3-540-95891-8_35.

[12] Gerald Lüttgen & Walter Vogler (2010): *Ready simulation for concurrency: It's logical!* *Inf. Comput.* 208(7), pp. 845–867, doi:10.1016/j.ic.2010.02.001.

[13] Nancy Lynch (1988): *I/O Automata: A model for discrete event systems*. *In 22nd Annual Conferenc e on Information Sciences and Systems* , pp. 29–38.
http://groups.csail.mit.edu/tds/papers/Lynch/MIT-LCS-TM-351.pdf

[14] R. Milner (1989): *Communication and Concurrency*. Prentice Hall.

[15] David Park (1981): *Concurrency and Automata on Infinite Sequences*. In *Theoretical Computer Science, 5th GI-Conference*, *LNCS* 104, Springer, pp. 167–183, doi:10.1007/BFb0017309.

Logics for Contravariant Simulations*

Ignacio Fábregas, David de Frutos Escrig, and Miguel Palomino

Departamento de Sistemas Informáticos y Computación, UCM
fabregas@fdi.ucm.es, {defrutos,miguelpt}@sip.ucm.es

Abstract. Covariant-contravariant simulation and conformance simulation are two generalizations of the simple notion of simulation which aim at capturing the fact that it is not always the case that "the larger the number of behaviors, the better". Therefore, they can be considered to be more adequate to express the fact that a system is a correct implementation of some specification. We have previously shown that these two more elaborated notions fit well within the categorical framework developed to study the notion of simulation in a generic way. Now we show that their behaviors have also simple and natural logical characterizations, though more elaborated than those for the plain simulation semantics.

1 Introduction and Some Related Work

Simulations are a very natural way to compare systems modeled by labeled transition systems or other related mechanisms based on describing the behavior of states by means of the actions they can execute [12]. They aim at comparing processes based on the simple premise "you are better if you can do as much as me, and perhaps some additional new things". This assumes that all the executable actions are controlled by the user (hence, no difference between input and output actions) and does not take into account that the system will choose in an unpredictable internal way whenever it has several possibilities for the execution of an action; thus, the more possibilities, the less control.

In order to cope with this situation one should consider adequate versions of simulation where the meaning of actions and the idea of preferring processes that are less non-deterministic are taken into account. This leads to two new notions of simulation: covariant-contravariant simulation and conformance simulation, that we roughly sketched in [6] and presented in detail in [7], where we proved that they can be obtained as particular instances of the general notion of categorical simulation developed by Hughes and Jacobs [9].

The first new notion is that of covariant-contravariant simulation, where the alphabet of actions Act is partitioned into three disjoint sets Act^l, Act^r, and Act^{bi}. The intention is that simulations will treat the actions in Act^l like in the ordinary case, they will interchange the roles of the related processes for those

* Research supported by the Spanish projects DESAFIOS10 TIN2009-14599-C03-01, TESIS TIN2009-14321-C02-01 and PROMETIDOS S2009/TIC-1465.

J. Hatcliff and E. Zucca (Eds.): FMOODS/FORTE 2010, LNCS 6117, pp. 224–231, 2010.

actions in Act^r, and they will impose a symmetric condition (like that defining bisimulation) for the actions in Act^{bi}. The second notion, conformance simulation, captures the conformance relations [10] that several authors introduced in order to formalize the notion of possible implementations.

After showing in [7] that they can be formalized as categorical simulations, in this paper we present their logical characterizations. We expect that they will contribute to clarify the meaning of the corresponding simulations, shedding light on the properties that can be established when using these two frameworks within a specification procedure.

Certainly, the distinction between input and output actions or similar classifications is not meant to be new at all and, for instance, it was present in modal transition systems as early as the end of the eighties. It also plays a central role in I/O-automata [11] and, more recently, appears as component of several works on interface automata [4], where the covariant-contravariant distinction is found when the guarantees of the specification can only be assumed if the conditions of the specification are satisfied.

Concerning conformance simulation, the first related references are also quite old [10] and correspond to the notion of conformance testing, which is close to failure semantics [13]. However, it is a bit surprising that in both cases there is lack of a basic theory where these notions are presented in a simplified scenario, stressing their main characteristics and properties.

Let us conclude this introduction by remarking that there is a large collection of recent papers where notions close to those studied here are either developed or applied. We regret not having the time or space to discuss, or even to cite, many of them and just to give a hint we point out [1,2], where several references to other preliminary works in those directions can be found.

2 Recalling Contravariant Simulations

We consider labeled transition systems (LTS) $(P, A, \rightarrow_P)$, where $\rightarrow_P \subseteq P \times A \times P$, to define the operational semantics of a family of processes $p \in P$. We say that the LTS is *finitary* when for each $p \in P$ and $a \in A$ we have $|\{p' \mid p \xrightarrow{a} p'\}| < \infty$.

We refer to [7] for a more extensive motivation of covariant-contravariant simulations; here we only comment on the case of input/output automata. To define an adequate simulation notion for them we observe that the classic approach to simulations is based on the definition of semantics for reactive systems, where all the actions of the processes correspond to input actions that the user must trigger. Instead, the situation is the opposite whenever we have explicit output actions: it is the system that produces the actions and the user who is forced to accept the produced output. Then, it is natural to conclude that in the simulation framework we have to dualize the simulation condition when considering output actions, and this is exactly what our anti-simulation relations do.

Definition 1. *Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$, two labeled transition systems for the alphabet A, and $\{A^r, A^l, A^{bi}\}$ a partition of this alphabet,*

a (A^r, A^l)***-simulation*** *(or just a covariant-contravariant simulation) between them is a relation* $S \subseteq P \times Q$ *such that for every* pSq *we have:*

- *for all* $a \in A^r \cup A^{bi}$ *and all* $p \xrightarrow{a} p'$ *there exists* $q \xrightarrow{a} q'$ *with* $p'Sq'$.
- *for all* $a \in A^l \cup A^{bi}$, *and all* $q \xrightarrow{a} q'$ *there exists* $p \xrightarrow{a} p'$ *with* $p'Sq'$.

We will write $p \lesssim_{CC} q$ *if there exists a covariant-contravariant simulation* S *such that* pSq.

Conformance simulations allow the extension of the set of actions offered by a process, so that in particular $a \lesssim a+b$, but they also consider that a process can be "improved" by reducing the nondeterminism in it, so that $ap+aq \lesssim ap$. In this way we have again a kind of covariant-contravariant simulation, not driven by the alphabet of actions executed by the processes but by their nondeterminism.

Definition 2. *Given* $P = (P, A, \rightarrow_P)$ *and* $Q = (Q, A, \rightarrow_Q)$ *two labeled transition systems for the alphabet* A, *a* ***conformance simulation*** *between them is a relation* $R \subseteq P \times Q$ *such that whenever* pRq, *then:*

- *For all* $a \in A$, *if* $p \xrightarrow{a}$, *then* $q \xrightarrow{a}$ *(this means, using the usual notation for process algebras, that* $I(p) \subseteq I(q)$*).*
- *For all* $a \in A$ *such that* $q \xrightarrow{a} q'$ *and* $p \xrightarrow{a}$, *there exists some* p' *with* $p \xrightarrow{a} p'$ *and* $p'Rq'$.

We will write $p \lesssim_{CS} q$ *if there exists a conformance simulation* R *such that* pRq.

3 Logical Characterizations of the New Semantics

3.1 Covariant-Contravariant Simulations

The class $\mathcal{L}_S$ characterizing the simulation semantics is defined in [3] as that containing $\mathtt{tt}$, conjunctions $\bigwedge_{i\in I} \varphi_i$ (which can be just finite or binary if we only want to characterize finitary process) and the existential operator $\langle a\rangle\varphi$, whose semantics is defined by: $p \models \langle a\rangle\varphi$ if there exists some p' such that $p \xrightarrow{a} p'$ and $p' \models \varphi$.

If we compare it with the Hennessy-Milner logic $\mathcal{L}_{HM}$ [8], it can be noted that the main diference is that negation is not present. Obviously, this must be the case to capture a strict order that is not an equivalence relation, such as $\lesssim_{CC}$. However, adding both the constant $\mathtt{ff}$ and the disjunction $\bigvee_{i\in I} \varphi_i$ does no harm, thus obtaining $\bar{\mathcal{L}_S}$ which also characterizes $\lesssim_S$. Indeed, $\mathtt{ff}$ is just $\bigvee_{\emptyset} \varphi_i$, while disjunctions can be moved to the top of the expression because $\langle a\rangle \bigvee_{i\in I} \varphi_i \equiv \bigvee_{i\in I}\langle a\rangle\varphi_i$, and $p \models \bigvee_{i\in I} \varphi_i$ iff there exists some $i \in I$ such that $p \models \varphi_i$.

The inspiration to obtain the logic characterizing $\lesssim_{CC}$ comes from the fact that if we only have contravariant actions, then $\lesssim_{CC}$ becomes $\lesssim_S^{-1}$, and therefore by negating all the formulas in $\bar{\mathcal{L}_S}$ we would obtain the desired characterization. In particular, for the modal operator $\langle a\rangle$ we would obtain its dual form $[a]$, whose semantics is defined by: $p \models [a]\varphi$ *if* $p' \models \varphi$ *for all* p' *such that* $p \xrightarrow{a} p'$.

Then, in the presence of both covariant and contravariant actions, we need to consider the existential operator $\langle a\rangle$ for $a \in A^r \cup A^{bi}$ and the universal operator $[a]$ for $a \in A^l \cup A^{bi}$, thus obtaining the following definition.

Definition 3. *Given an alphabet A, and $\{A^r, A^l, A^{bi}\}$ a partition of this alphabet, the class $\mathcal{L}_{CC}$ of covariant-contravariant simulation formulas over A is defined recursively by:*

- $\mathtt{tt}$ *and* $\mathtt{ff}$ *are in* $\mathcal{L}_{CC}$.
- *If I is a set and $\varphi_i \in \mathcal{L}_{CC}$ for all $i \in I$ then $\bigwedge_{i\in I} \varphi_i \in \mathcal{L}_{CC}$, $\bigvee_{i\in I} \varphi_i \in \mathcal{L}_{CC}$.*
- *If $\varphi \in \mathcal{L}_{CC}$ and $a \in A^r \cup A^{bi}$ then $\langle a\rangle\varphi \in \mathcal{L}_{CC}$.*
- *If $\varphi \in \mathcal{L}_{CC}$ and $a \in A^l \cup A^{bi}$ then $[a]\varphi \in \mathcal{L}_{CC}$.*

The satisfaction relation $\models$ is defined recursively by:

- $p \models \mathtt{tt}$.
- $p \models \bigwedge_{i\in I} \varphi_i$ *if* $p \models \varphi_i$ *for all* $i \in I$.
- $p \models \bigvee_{i\in I} \varphi_i$ *if* $p \models \varphi_i$ *for some* $i \in I$.
- $p \models \langle a\rangle\varphi$ *if there exists some* p' *such that* $p \xrightarrow{a} p'$ *and* $p' \models \varphi$.
- $p \models [a]\varphi$ *if* $p' \models \varphi$ *for all* p' *such that* $p \xrightarrow{a} p'$.

Let $\mathcal{S}_{CC}(p)$ denote the class of covariant-contravariant simulation formulas satisfied by the process p, that is, $\mathcal{S}_{CC}(p) = \{\varphi \in \mathcal{L}_{CC} \mid p \models \varphi\}$. We will write $p \preceq_{CC} q$ if $\mathcal{S}_{CC}(p) \subseteq \mathcal{S}_{CC}(q)$.

The case of input/output transition systems is probably the clearest example where the covariant-contravariant duality must be applied in order to capture the appropriate simulation order. Input actions should have a covariant behavior reflecting the fact that a reactive system is expected to be "better" whenever it accepts a maximal set of requests; as a consequence, its logical characterization can only capture liveness properties. Conversely, output actions should be contravariant: whenever we specify a system we expect to control its behavior as much as possible, and outputs are generative, which means not controllable by the user. This contravariant character is captured by the universal operator $[a]$, which is only able to define safety properties.

Therefore, the logic $\mathcal{L}_{CC}$ includes formulas that simultaneously capture liveness and safety at a local level, depending on the character of the actions that are used. This is not enough to adequately state all the requirements one could possible need: certainly, after developing a myriad of different semantics for processes [13,5], we would not expect that just by fiddling with one of the simplest, the simulation semantics, we would have the definite answer to treat together covariant and contravariant actions. We are also investigating the covariant-contravariant version of other semantics but, in order to establish which are the basic facts to take into account, it is clear to us that the case of plain simulation is definitely a basic keystone.

Proposition 1. $p \lesssim_{CC} q \iff p \preceq_{CC} q$.

Proof. We will first prove the implication from left to right. Assume that we have pSq for some covariant-contravariant simulation S: we must show that for each $\varphi \in \mathcal{L}_{CC}$, $p \models \varphi$ implies $q \models \varphi$. We proceed by structural induction over φ.

- $q \models \mathsf{tt}$, trivially.
- Let $p \models \langle a \rangle \varphi$ with $a \in A^r \cup A^{bi}$. Then there is p' such that $p \xrightarrow{a} p'$ with $p' \models \varphi$. Now, since pRq and $a \in A^r \cup A^{bi}$ there must be a q' such that $q \xrightarrow{a} q'$ with $p'Rq'$ and, by induction hypothesis, $q' \models \varphi$, that is, $q \models \langle a \rangle \varphi$.
- Let $p \models [a]\varphi$. Then for all p' such that $p \xrightarrow{a} p'$ we have $p' \models \varphi$. Let q' be such that $q \xrightarrow{a} q'$ then, since pSq and $a \in A^l \cup A^{bi}$, there exists p' such that $p \xrightarrow{a} p'$ and $p'Sq'$. By induction hypothesis, since $p' \models \varphi$ then $q' \models \varphi$, that is, $q \models [a]\varphi$.
- Let $p \models \bigwedge_{i \in I} \varphi_i$. Then $p \models \varphi_i$ for all $i \in I$, so by induction hypothesis $q \models \varphi_i$ for all $i \in I$ and then $q \models \bigwedge_{i \in I} \varphi_i$.
- $p \models \bigvee_{i \in I} \varphi_i$. It is analogous to the previous case.

For the other implication let us assume that $p \preceq_{CC} q$ and show that $\preceq_{CC}$ is a covariant-contravariant simulation. Let $a \in A^r \cup A^{bi}$ and $p \xrightarrow{a} p'$; then there exists q' such that $q \xrightarrow{a} q'$ and $p' \preceq_{CC} q'$. Otherwise, we have that for all $q \xrightarrow{a} q'$, $p' \not\preceq_{CC} q'$, that is, we have formulas $\varphi_{q'}$ such that $\varphi_{q'} \in \mathcal{S}_{CC}(p') \setminus \mathcal{S}_{CC}(q')$. Now, taking $\phi = \langle a \rangle \bigwedge_{q'} \varphi_{q'}$, we have $p \models \phi$ and, by hypothesis, also $q \models \phi$. That means that there exists some q'_0 such that $q \xrightarrow{a} q'_0$ with $q'_0 \models \bigwedge_{q'} \varphi'_q$. But this cannot be the case since $q'_0 \not\models \varphi_{q'_0}$.

Now let $a \in A^l \cup A^{bi}$ and $q \xrightarrow{a} q'$; similarly we must show that there exists p' such that $p \xrightarrow{a} p'$ and $p' \preceq_{CC} q'$. By way of contradiction, if for all $p \xrightarrow{a} p'$ we have $p' \not\preceq_{CC} q'$, there are formulas $\varphi_{p'} \in \mathcal{S}_{CC}(p') \setminus \mathcal{S}_{CC}(q')$. Taking $\phi = [a] \bigvee_{p'} \varphi_{p'}$ we have $p \models \phi$ and then by hypothesis $q \not\models \phi$, but this cannot be since $q' \not\models \varphi_{p'}$ for all p'. □

3.2 Conformance Simulations

Conformance simulation can be considered to be a variant of the covariant-contravariant framework in which, instead of separating the actions in several classes, we have a mixed uniform behavior for all the actions. This is brought forward by the fact that if a process cannot execute a, then $p \lesssim_{CS} p + aq$. However, once we have $a \in I(p)$ the contravariant character shows since then $p + aq \lesssim_{CS} p$.

This mixed character of all the actions is now captured at the logical level by a new modal operator a, whose semantics is defined by: $p \models a\varphi$ *if* $p \xrightarrow{a}$ *and* $p' \models \varphi$ *for all* $p \xrightarrow{a} p'$. It is quite interesting to observe that we can alternatively define a as "$\langle a \rangle \wedge [a]$", since we have: $p \models a\varphi \iff p \models \langle a \rangle \varphi$ *and* $p \models [a]\varphi$, which also reveals the mixed intended nature of all the actions in the conformance framework.

Definition 4. *The class $\mathcal{L}_{CS}$ of conformance simulation formulas over A is defined recursively by:*

- $\mathsf{tt} \in \mathcal{L}_{CS}$.
- *If I is a set and $\varphi_i \in \mathcal{L}_{CS}$ for all $i \in I$ then $\bigwedge_{i \in I} \varphi_i, \in \mathcal{L}_{CS}$, $\bigvee_{i \in I} \varphi_i \in \mathcal{L}_{CS}$.*
- *If $\varphi \in \mathcal{L}_{CS}$ and $a \in A$ then $a\varphi \in \mathcal{L}_{CS}$.*

The corresponding satisfaction relation $\models$ *is defined recursively by:*

- $p \models \mathtt{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ *if* $p \models \varphi_i$ *for all* $i \in I$.
- $p \models \bigvee_{i \in I} \varphi_i$ *if* $p \models \varphi_i$ *for some* $i \in I$.
- $p \models a\varphi$ *if* $p \xrightarrow{a}$ *and* $p' \models \varphi$ *for all* $p \xrightarrow{a} p'$.

Let $\mathcal{S}_{CS}(p)$ *denote the class of conformance simulation formulas satisfied by the process* p*, that is,* $\mathcal{S}_{CS}(p) = \{\varphi \in \mathcal{L}_{CS} \mid p \models \varphi\}$*. We will write* $p \preceq_{CS} q$ *if* $\mathcal{S}_{CS}(p) \subseteq \mathcal{S}_{CS}(q)$.

One now expects that the liveness and safety requirements will be captured simultaneously and this is indeed the case since from $p \models a\varphi$ we know both that p is able to execute a and that, after executing it in any possible way, φ will be satisfied. Therefore, conformance simulation proves to be quite a reasonable semantics whenever we do not want to distinguish between reactive and generative actions, as discussed in the previous section.

Proposition 2. $p \lesssim_{CS} q \iff p \preceq_{CS} q$.

Proof. We first prove the implication from left to right. Assume that we have pRq for some conformance simulation R: we must show that for each $\varphi \in \mathcal{L}_{CS}$, $p \models \varphi$ implies $q \models \varphi$. The proof will follow by structural induction over φ, the case for $\mathtt{tt}$ being trivial.

- Let $p \models a\varphi$. Then, for all $p \xrightarrow{a} p'$ we have $p' \models \varphi$ and there exists at least one such p'. Since pRq also $q \xrightarrow{a}$, and it remains to prove that $q' \models \varphi$ for all successors $q \xrightarrow{a} q'$. Let q'_0 be such that $q \xrightarrow{a} q'_0$. Again, since pRq and $p \xrightarrow{a}$, for each $q \xrightarrow{a} q'$ there exists some $p \xrightarrow{a} p'$ such that $p'Rq'$. So, for q'_0 there exists p'_0 such that $p'_0 R q'_0$ and, since $p'_0 \models \varphi$, by induction hypothesis also $q'_0 \models \varphi$. Thus $q \models a\varphi$.
- Let $p \models \bigwedge_{i \in I} \varphi_i$. Then $p \models \varphi_i$ for all $i \in I$, so by induction hypothesis $q \models \varphi_i$ for all $i \in I$ and then $q \models \bigwedge_{i \in I} \varphi_i$.
- $p \models \bigvee_{i \in I} \varphi_i$. It is analogous to the previous case.

For the other implication, let us assume that $p \preceq_{CS} q$: we show that $\preceq_{CS}$ is a conformance simulation. First, if $p \xrightarrow{a}$ then, since $\mathcal{S}_{CS}(p) \subseteq \mathcal{S}_{CS}(q)$ and $p \models a\mathtt{tt}$, also $q \models a\mathtt{tt}$ and hence $q \xrightarrow{a}$. Now, let $q \xrightarrow{a} q'$ and $p \xrightarrow{a}$. Let us see that there exists some p' such that $p \xrightarrow{a} p'$ and $p' \preceq_{CS} q'$. By way of contradiction, if $p' \not\preceq_{CS} q'$ for all such p', then for each p' there is a formula $\varphi_{p'} \in \mathcal{S}_{CS}(p') \setminus \mathcal{S}_{CS}(q')$. Let $\phi = a \bigvee_{p'} \varphi_{p'}$. It is easy to see that $p \models \phi$: indeed, for each p' such that $p \xrightarrow{a} p'$, $p' \models \varphi_{p'}$. Since $p \preceq_{CS} q$, it must also be the case that $q \models \phi$, that is, for each q'' such that $q \xrightarrow{a} q''$, $q'' \models \bigvee_{p'} \varphi_{p'}$; but $q \xrightarrow{a} q'$ and $q' \not\models \varphi_{p'}$ for any p', contradicting the fact that $q \models \phi$. □

4 Some Examples and a Short Discussion

We will start by illustrating the behavior of covariant-contravariant simulations in the case in which we distinguish between input (reactive) and output (generative) actions. Consider the following expending machines:

$$\begin{array}{ll} \mathsf{onecoke} & : \mathsf{coin} \to \mathsf{coke} \to 0 \\ \mathsf{cokeorlemonade} & : \mathsf{coin} \to \big((\mathsf{coke} \to 0) + (\mathsf{lemonade} \to 0)\big) \end{array}$$

The classical approach would consider $\mathsf{onecoke} \lesssim_S \mathsf{cokeorlemonade}$. However, if the drinks are provided by the machine in an autonomous way then they should be formalized as outputs, which leads us to

$$\mathsf{cokeorlemonade} \lesssim_{CC} \mathsf{onecoke}.$$

This is justified by the fact that choices between generative actions become internal and therefore generate (undesired) non-deterministic behavior.

At the logical level the difference between the two processes above can be brought forward by means of the formula $\langle \mathsf{coin} \rangle\ [\mathsf{lemonade}]\ \mathsf{ff}$, which onecoke satisfies but cokeorlemonade does not. It could be thought that the process cokeorlemonade is being punished for offering lemonade besides coke, but this would be an incorrect interpretation because it follows the classical reactive approach where simultaneous offers mean "the user makes his choice"; instead, when outputs are generative it is the machine that chooses. As a consequence, from $\mathsf{cokeorlemonade} \not\models \langle \mathsf{coin} \rangle [\mathsf{lemonade}]\ \mathsf{ff}$ we implicitly infer that it could be the case that after inserting a coin we did not get our favorite drink (Coke).

Let us now show the differences between covariant-contravariant and conformance simulations. First, at the formal level, the fact that the modal operator a can be defined as "$\langle a \rangle \wedge [a]$" does not mean that these two basic modal operators can appear separately in a formula characterizing $\lesssim_{CS}$. Obviously this cannot be the case since separated $\langle a \rangle$ operators characterize plain simulation, and for the process $\mathsf{choice\_coke\_lemonade}$: $(\mathsf{coin} \to \mathsf{coke} \to 0) + (\mathsf{coin} \to \mathsf{lemonade} \to 0)$ we have

$$\mathsf{choice\_coke\_lemonade} \models \langle \mathsf{coin} \rangle \langle \mathsf{lemonade} \rangle \mathsf{tt} \qquad \mathsf{onecoke} \not\models \langle \mathsf{coin} \rangle \langle \mathsf{lemonade} \rangle \mathsf{tt}$$

but $\mathsf{choice\_coke\_lemonade} \lesssim_{CS} \mathsf{onecoke}$.

Now, if we consider de universal operator $[a]$, its weakness when used alone arises when it is trivially satisfied. For instance, we have $0 \models [\mathsf{coin}]\ \mathsf{ff}$ but $\mathsf{onecoke} \not\models [\mathsf{coin}]\ \mathsf{ff}$ and $0 \lesssim_{CS} \mathsf{onecoke}$.

One could infer that conformance simulation is the definitive solution to capture all the natural requirements in an specification. Certainly, it combines covariant and contravariant aspects in a very balanced way, but the fact that it treats all the actions uniformly makes it impossible to capture the difference between input and output actions. In particular: $\mathsf{onecoke} \lesssim_{CS} \mathsf{cokeorlemonade}$ but we have already discussed that when outputs are generative, choices always generate non-deterministic behaviors that $\lesssim_{CS}$ is not punishing at all.

On the other hand, choices between equal actions are also considered "harmful" by the conformance semantics so that if $p \lesssim_{CS} q$ then $ap =_{CS} ap + aq$. This is sometimes a too pessimistic approach, which we can illustrate by the following slot_machine specification:

$$\mathsf{slot\_machine} : (\mathsf{coin} \to \mathsf{souvenir} \to 0) + (\mathsf{coin} \to ((\mathsf{million\$} \to 0) + (\mathsf{souvenir} \to 0)))$$

which becomes conformance simulation equivalent to the pluff_machine

$$\mathsf{pluff\_machine} : \mathsf{coin} \to \mathsf{souvenir} \to 0$$

In this case the possible return of the big pot is not taken into account at all. Obviously, the solution comes from choosing in each case the adequate semantics to capture accurately the desired behaviors. The bad news is that we need to study many diferent semantics; the good news for us is... the same!, since we are already working on them

References

1. Antonik, A., Huth, M., Larsen, K., Nyman, U., Wasowski, A.: 20 Years of Mixed and Modal Specifications. Bulletin of the European Association for Theor. Comput. Sci. (May 2008)
2. Benes, N., Kretínský, J., Larsen, K.G., Srba, J.: On determinism in modal transition systems. Theor. Comput. Sci. 410(41), 4026–4043 (2009)
3. Cîrstea, C.: A modular approach to defining and characterising notions of simulation. Inf. Comput. 204(4), 469–502 (2006)
4. de Alfaro, L., Henzinger, T.A.: Interface automata. In: ESEC / SIGSOFT FSE, pp. 109–120 (2001)
5. de Frutos Escrig, D., Gregorio-Rodríguez, C., Palomino, M.: On the unification of semantics for processes: observational semantics. In: Nielsen, M., Kucera, A., Miltersen, P.B., Palamidessi, C., Tuma, P., Valencia, F.D. (eds.) SOFSEM 2009. LNCS, vol. 5404, pp. 279–290. Springer, Heidelberg (2009)
6. de Frutos-Escrig, D., Rosa Velardo, F., Gregorio-Rodríguez, C.: New bisimulation semantics for distributed systems. In: Derrick, J., Vain, J. (eds.) FORTE 2007. LNCS, vol. 4574, pp. 143–159. Springer, Heidelberg (2007)
7. Fábregas, I., de Frutos-Escrig, D., Palomino, M.: Non-strongly stable orders also define interesting simulation relations. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO 2009. LNCS, vol. 5728, pp. 221–235. Springer, Heidelberg (2009)
8. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. J. ACM 32(1), 137–161 (1985)
9. Hughes, J., Jacobs, B.: Simulations in coalgebra. Theor. Comput. Sci. 327(1-2), 71–108 (2004)
10. Leduc, G.: A framework based on implementation relations for implementing LOTOS specifications. Computer Networks and ISDN Systems 25(1), 23–41 (1992)
11. Lynch, N.: I/O automata: A model for discrete event systems. In: 22nd Annual Conference on Information Sciences and Systems, pp. 29–38 (1988)
12. Park, D.: Concurrency and automata on infinite sequences. In: GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
13. van Glabbeek, R.J.: The linear time-branching time spectrum I: The semantics of concrete, sequential processes. In: Handbook of process algebra, pp. 3–99. North-Holland, Amsterdam (2001)

Equational Characterization of Covariant-Contravariant Simulation and Conformance Simulation Semantics

Ignacio Fábregas David de Frutos Escrig Miguel Palomino
Universidad Complutense de Madrid
Madrid, Spain
Departamento de Sistemas Informáticos y Computación *
fabregas@fdi.ucm.es defrutos@sip.ucm.es miguelpt@sip.ucm.es

Covariant-contravariant simulation and conformance simulation generalize plain simulation and try to capture the fact that it is not always the case that "the larger the number of behaviors, the better". We have previously studied their logical characterizations and in this paper we present the axiomatizations of the preorders defined by the new simulation relations and their induced equivalences. The interest of our results lies in the fact that the axiomatizations help us to know the new simulations better, understanding in particular the role of the contravariant characteristics and their interplay with the covariant ones; moreover, the axiomatizations provide us with a powerful tool to (algebraically) prove results of the corresponding semantics. But we also consider our results interesting from a metatheoretical point of view: the fact that the covariant-contravariant simulation equivalence is indeed ground axiomatizable when there is no action that exhibits both a covariant and a contravariant behaviour, but becomes non-axiomatizable whenever we have together actions of that kind and either covariant or contravariant actions, offers us a new subtle example of the narrow border separating axiomatizable and non-axiomatizable semantics. We expect that by studying these examples we will be able to develop a general theory separating axiomatizable and non-axiomatizable semantics.

1 Introduction and some related work

Simulations are a very natural way to compare systems defined by labeled transition systems or other related mechanisms based on describing the behavior of states by means of the actions they can execute [19]. They aim at comparing processes based on the simple premise "you are better if you can do as much as me, and perhaps some other new things". This assumes that all the executable actions are controlled by the user (no difference between input and output actions) and does not take into account that whenever the system has several possibilities for the execution of an action it will choose in an unpredictable internal way, so that more possibilities means less control.

In order to cope with these limitations one should consider adequate versions of simulation where the characteristics of actions and the idea of preferring processes that are less non-deterministic are taken into account. This leads to two new notions of simulation: covariant-contravariant simulation and conformance simulation that we roughly sketched in [10] and presented in detail in [12], where we proved that they can be presented as particular instances of the general notion of categorical simulation developed by Hughes and Jacobs [14].

Certainly, the distinction between input and output actions or similar classifications is not meant to be new at all and, for instance, they were present in modal transition systems as early as the end of the

*Research supported by the Spanish projects DESAFIOS10 TIN2009-14599-C03-01, TESIS TIN2009-14321-C02-01 and PROMETIDOS S2009/TIC-1465. The second author worked in this paper during a visit to Reykjavik University sponsored through a grant by the ABEL Extraordinary Chair.

L. Aceto, P. Sobociński (Eds.): Seventh Workshop on
Structural Operational Semantics 2010 (SOS '10)
EPTCS 32, 2010, pp. 1–14, doi:10.4204/EPTCS.32.1

eighties. They also play a central role in I/O-automata [18] and more recently appear as component of several works on interface automata [7, 15], where one finds the covariant-contravariant distinction when the guarantees of the specification can only be assumed if the conditions of the specification are satisfied.

Concerning conformance simulation, the first related references are also quite old [17, 21], corresponding to the notion of conformance testing, which is close to failure semantics [4]. However, it is a bit surprising that in both cases we lack a basic theory where these notions are presented in a simplified scenario, stressing their main characteristics and properties. We think that the theory of semantics for processes, and particularly the simulation semantics, is a perfect field in which to develop that basic theory. This has been already proved in [12], where our new simulation semantics were shown to be categorical simulations, thus inheriting all their good properties for free.

In [11] we have also briefly presented the logical characterizations of the two semantics. Now that we already know quite well the behaviour of the two new notions of simulation we can give their algebraic presentation. By the way, although in our previous works on the unified study of process semantics the (classical) covariant character of all the actions had several important consequences, mainly represented by the extremely simple and easy to apply basic axiom for simulation $(\mathrm{S})\ x \sqsubseteq x+y$ (or equivalently, just $0 \sqsubseteq y$), we have been able to borrow from [9, 1, 8] several ideas about the axiomatization of process semantics that, although not directly applicable due to the special characteristics of the new semantics, can be adequately adapted.

However, not all of the simple and nice results for the algebraic theory of plain (covariant) simulation can be extended to the general covariant-contravariant case. In particular, in order to obtain the maximal genericity, when we defined covariant-contravariant simulations in [12] we admitted not only both covariant and contravariant actions, but also other actions with a bivariant nature. This decision was taken because when presenting a general theory of categorical simulations in [14], J. Hughes and B. Jacobs already noticed that bisimulation was a particular (in fact, trivial) example of simulation semantics. It was also clear that inverse simulation (namely, contravariant simulation) was also another example, and then we were able to prove that our general covariant-contravariant simulation was another categorical simulation that smoothly combines bisimulation, plain (covariant) simulation and inverse (contravariant) simulation.

Obviously, plain bisimulation has a simple axiomatization, as is the case for plain simulation; we will see in this paper that the preorder defined by our covariant-contravariant simulation can also be finitely axiomatized. When we considered the induced equivalence, we found indeed a finite axiomatization for the case in which there are no bivariant actions (actions that can be considered as both input and output) in our alphabet. The axiomatization and its completeness proof were obtained by adapting the general techniques in [8, 9] for the covariant case to our more general covariant-contravariant scenario. However, as soon as a single bivariant action is introduced, and at least one non-bivariant one is also present, then the equational theory of covariant-contravariant simulation equivalence becomes non-finitely axiomatizable, and in fact the proof of this result is extraordinarily simple.

Even if this is a negative result, we think that it will contribute to enlight the narrow border separating axiomatizable and non-axiomatizable process theories, which we expect to continue exploring in the future.

There is a large collection of recent papers where notions close to those studied here are either developed or applied; a detailed comparison will appear elsewhere. However, we insist on the fact that we were not able to find a basic study where the main results on process theory had been extended to a framework containing any contravariant characteristics, although it is true that some small contributions along this direction can be found in some of these papers. We plan to develop a thorough compilation of the works on this topic by isolating the places where our foundational study could help to understand

the different developments, as well as looking for applications and new enhancements to our theory that could be of use to relate all the disconnected work on the area. In turn, we hope that this will also provide us with some intuition to understand those results and produce new formal techniques to obtain proofs of those, or other interesting results in the area. So, simply to give a hint, a sample of those works would include [2, 3, 16, 20].

2 Preliminaries

In this section we summarize some definitions and concepts from [6, 12] and introduce the notation we are going to use.

Let us recall our two new simulation notions:

Definition 1 *Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$, two labeled transition systems (LTS) for the alphabet A, and $\{A^r, A^l, A^{bi}\}$ a partition of this alphabet, a (A^r, A^l)-**simulation** (or just a covariant-contravariant simulation) between them is a relation $S \subseteq P \times Q$ such that for every pSq we have:*

- *For all $a \in A^r \cup A^{bi}$ and all $p \xrightarrow{a} p'$ there exists $q \xrightarrow{a} q'$ with $p'Sq'$.*
- *For all $a \in A^l \cup A^{bi}$, and all $q \xrightarrow{a} q'$ there exists $p \xrightarrow{a} p'$ with $p'Sq'$.*

We will write $p \lesssim_{CC} q$ if there exists a covariant-contravariant simulation S such that pSq.

This definition combines the requirements of plain simulation, for some of the actions, with those of plain "anti-simulation", for some of the remaining actions, imposing both on so-called bivariant actions.

Definition 2 *Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$ two labeled transition systems for the alphabet A, a **conformance simulation** between them is a relation $R \subseteq P \times Q$ such that whenever pRq, then:*

- *For all $a \in A$, if $p \xrightarrow{a}$, then $q \xrightarrow{a}$ (this means, using the usual notation for process algebras, that $I(p) \subseteq I(q)$).*
- *For all $a \in A$ such that $q \xrightarrow{a} q'$ and $p \xrightarrow{a}$, there exists some p' with $p \xrightarrow{a} p'$ and $p'Rq'$.*

We will write $p \lesssim_{CS} q$ if there exists a conformance simulation R such that pRq.

The first clause of the definition guarantees that Q has at least all the behaviors of P, allowing to "improve" a process by extending the set of actions it offers, whereas the second clause establishes that a process can be "improved" by reducing the nondeterminism in it.

Let us recall that the set *BCCSP(A)* of basic processes for the alphabet A is defined by the *BNF*-grammar

$$p ::= 0 \mid ap \mid p + p$$

where $a \in A$. The operational semantics for BCCSP terms is defined by

$$ap \xrightarrow{a} p \qquad \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

With these operators we can only define finite processes; however, it is well known that these operators capture the essence of any transition system, which can be defined by a system of equations specifying the behavior of each state. (The axioms for recursive processes, other interesting extensions including the communication operators, and possibly some others, are left for future work.)

3 Axiomatization of the new simulation preorders

In this section we present a finite axiomatization of the two preorders for basic finite processes induced by our new kinds of simulation.

3.1 Covariant-contravariant semantics

We consider a partition $\{A^r, A^l, A^{bi}\}$ of the alphabet A, with actions that have either a covariant nature, or contravariant, or both at the same time. Contravariant simulation $\lesssim_S^{-1}$ is just the inverse of plain simulation and therefore can be trivially axiomatized by inverting the axiom for plain simulation

$(\mathsf{S})\quad x \sqsubseteq x+y,$

thus obtaining

$(\mathsf{S}^{-1})\quad x+y \sqsubseteq x.$

In order to produce an axiomatization of covariant-contravariant simulation we need to combine in an adequate way these two axioms, by constraining each of them to the case in which the added process y only offers actions with the corresponding covariant or contravariant character. Hence we obtain:

$(\mathsf{S}^r)\quad I(y) \subseteq A^r \Longrightarrow x \sqsubseteq x+y.$

$(\mathsf{S}^{-1,l})\quad I(y) \subseteq A^l \Longrightarrow x+y \sqsubseteq x.$

We can omit the conditions in these two axioms by considering two generic actions $a_r \in A^r$ and $a_l \in A^l$:

$(\mathsf{S}^r_p)\quad x \sqsubseteq x+a_r y.$

$(\mathsf{S}^l_p)\quad x+a_l y \sqsubseteq x.$

Note that actions in A^{bi} do not appear in the axioms above, although they could be included in the processes instantiating the variables x and y. This is an immediate consequence of the fact that their behavior corresponds to that governed by bisimulation, so that we need not add any new axiom to those capturing the bisimilarity relation:

$(\mathsf{B}_1)\quad x+y = y+x.$

$(\mathsf{B}_2)\quad (x+y)+z = x+(y+z).$

$(\mathsf{B}_3)\quad x+x = x.$

$(\mathsf{B}_4)\quad x+0 = x.$

We will use these axioms implicitly in the remainder of this paper.

Proposition 1 *The (A^r, A^l)-simulation preorder can be axiomatically defined by means of the set of axioms $\{\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3, \mathsf{B}_4, \mathsf{S}^r_p, \mathsf{S}^l_p\}$.*

Proof. First we prove that the axioms (S^r_p) and (S^l_p) are sound for the (A^r, A^l)-similarity relation $\lesssim_{CC}$. Indeed:

- For all $a \in A^r \cup A^{bi}$, if $x \xrightarrow{a} x'$ then $x+a_r y \xrightarrow{a} x'$ and $x' \lesssim_{CC} x'$.
- For all $a \in A^l \cup A^{bi}$, if $x+a_r y \xrightarrow{a} x'$, then $x \xrightarrow{a} x'$ and $x' \lesssim_{CC} x'$. Note that $a \neq a_r$ since $A^r \cap (A^l \cup A^{bi}) = \emptyset$.
- For all $a \in A^r \cup A^{bi}$, if $x+a_l y \xrightarrow{a} x'$ then $x \xrightarrow{a} x'$ and $x' \lesssim_{CC} x'$ as above, because $a \neq a_l$ again.

- For all $a \in A^l \cup A^{bi}$, if $x \xrightarrow{a} x'$, then $x + a_l y \xrightarrow{a} x'$ and $x' \lesssim_{CC} x'$.

To prove completeness we consider $p \lesssim_{CC} q$ and reason by structural induction on p.

- If p is 0 then $I(q) \subseteq A^r$, since p cannot simulate any action in $A^l \cup A^{bi}$. Then $q = \sum a_r q_r$ and we can apply (S^r_p) to each summand in turn to get $0 \sqsubseteq q$.
- Let us consider $p = (\sum a_r p_r + \sum a_l p_l + \sum a_b p_b)$, distinguishing the summands of p which start with actions in either A^r, A^l or A^{bi}. We decompose q in the same way to obtain $q = (\sum b_r q_r + \sum b_l q_l + \sum b_b q_b)$. Then:
  - For every a_r there exists b_r, with $a_r = b_r$, such that $p_r \lesssim_{CC} q_r$ and, by induction hypothesis, $p_r \sqsubseteq q_r$. Then $\sum a_r p_r \sqsubseteq \sum b_r q_r$. It could be the case that some summands of $\sum b_r q_r$ are never used to simulate any of the transitions of p, but then we can add all those summand by using (S^r_p), to derive $\sum a_r p_r \sqsubseteq \sum b_r q_r$.
  - For the summands $\sum a_l p_l$ and $\sum b_l q_l$ we can argue in exactly the same way, but starting with the righhand side and using (S^l_p) instead of (S^r_p), to conclude now $\sum a_l p_l \sqsubseteq \sum b_l q_l$.
  - Finally, using standard arguments for bisimulation, we can establish a full correspondence between the summands $\sum a_b p_b$ and $\sum b_b q_b$, having $a_b = b_b$ and $p_b \lesssim_{CC} q_b$, and by induction hypothesis we prove $\sum a_b p_b \sqsubseteq \sum b_b q_b$, thus concluding the proof. □

3.2 Conformance semantics

Conformance simulation combines in a curious manner the features of both ordinary (covariant) and inverse (contravariant) simulation: the addition of new capabilities is always considered beneficial but, when an action is already offered, new ways to execute it are avoided since this leads to a more non-deterministic process.

To capture the first situation we need a variant of the axiom (S) characterizing ordinary simulation:

$$(\mathsf{S}_{CS}) \quad I(p) \cap I(q) = \emptyset \Longrightarrow p \sqsubseteq p + q.$$

For the latter, we instantiate the axiom (S^{-1}) obtaining

$$(\mathsf{S}^{-1}_{CS}) \quad I(q) \subseteq I(p) \Longrightarrow p + q \sqsubseteq p,$$

which can be equivalently stated as

$$(\mathsf{S}^{-1}_{CS,p}) \quad ap + aq \sqsubseteq ap.$$

There is, however, an important drawback: conformance simulation is not a precongruence because it is not always preserved by $+$. Indeed, $0 \lesssim_{CS} ab$ and $ac \lesssim_{CS} ac$, but not $ac \lesssim_{CS} ab + ac$. Fortunately, to obtain a satisfactory algebraic treatment of the conformance order it is enough to consider the weakest precongruence contained in it, as is done for weak bisimulation and the corresponding observation congruence. Let us simply replace the axiom (S_{CS}) by its guarded version

$$(\mathsf{S}_{CS,g}) \quad I(p) \cap I(q) = \emptyset \Longrightarrow ap \sqsubseteq a(p + q).$$

Definition 3 *We define the conformance precongruence relation $p \lesssim^p_{CS} q$ by*

$$p \lesssim^p_{CS} q \iff (p \lesssim_{CS} q \text{ and } I(p) \supseteq I(q)).$$

Note that the condition $I(p) \supseteq I(q)$ is not imposed recursively but just on the initial states of the processes, which corresponds to the fact that the (once) guarded axiom $(\mathsf{S}_{CS,g})$ becomes sound for the classical substitution calculus, in order to characterize the conformance precongruence $\lesssim^p_{CS}$.

Proposition 2 *If the set of actions A is infinite, then the precongruence relation $\lesssim^p_{CS}$ is the coarsest precongruence contained in $\lesssim_{CS}$.*

Proof. Obviously, we have $\lesssim^p_{CS} \subseteq \lesssim_{CS}$. If there were a larger precongruence, there would exist p and q with $p \lesssim_{CS} q$ but $I(q) \not\subseteq I(p)$: then, taking $a \in I(q) \setminus I(p)$ and $b \in A$ such that $q \stackrel{a \cdot b}{\not\longrightarrow}$ we would have $ab + p \not\lesssim_{CS} ab + q$ (since $ab \not\lesssim_{CS} q$).

Finally, both the prefix operator and $+$ preserve $\lesssim^p_{CS}$:

- If $p \lesssim^p_{CS} q$, then $ap \lesssim^p_{CS} aq$ since $I(ap) = I(aq) = \{a\}$, and for $aq \stackrel{a}{\longrightarrow} q$ we have $ap \stackrel{a}{\longrightarrow} p$ with $p \lesssim^p_{CS} q$.
- If $p \lesssim^p_{CS} q$, then $ap + r \lesssim^p_{CS} aq + r$ since $I(ap + r) = I(aq + r) = I(r) \cup \{a\}$, and for $aq + r \stackrel{a}{\longrightarrow} q$ we have $ap + r \stackrel{a}{\longrightarrow} p$ with $p \lesssim^p_{CS} q$ and, whenever $aq + r \stackrel{b}{\longrightarrow} r'$ with $r \stackrel{b}{\longrightarrow} r'$, we trivially have $ap + r \stackrel{b}{\longrightarrow} r'$. □

Proposition 3 *The set of axioms $\mathscr{A}_{CS} = \{\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3, \mathsf{B}_4, \mathsf{S}_{CS,g}, \mathsf{S}^{-1}_{CS,p}\}$ is complete for the conformance precongruence relation $\lesssim^p_{CS}$.*

Proof. We show by induction on the depth of p that, whenever $p \lesssim^p_{CS} q$ (resp. $bp \lesssim^p_{CS} bq$), we have $\mathscr{A}_{CS} \vdash p \sqsubseteq q$ (resp. $\mathscr{A}_{CS} \vdash bp \sqsubseteq bq$).

- If $0 \lesssim^p_{CS} q$, then also $q = 0$ and $0 \sqsubseteq 0$ using $(\mathsf{S}^{-1}_{CS,p})$.
- If $b0 \lesssim^p_{CS} bq$, then we can apply $(\mathsf{S}_{CS,g})$ with $p = 0$.

Let us now consider $p = \sum_{a_i \in I(p)} a_i p_{ij}$ and $q = \sum_{a_i \in I(q)} a_i q_{ik}$.

- If $p \lesssim^p_{CS} q$ then $I(p) = I(q)$ and $p \lesssim_{CS} q$, so for each q_{ik} there is some p_{ij} with $p_{ij} \lesssim_{CS} q_{ik}$ and therefore we can apply the second induction hypothesis to conclude that $a_i p_{ij} \sqsubseteq a_i q_{ik}$. It is possible that some summands p_{ij} will be paired with no q_{ik} in the step above, but then we can apply the axiom $(\mathsf{S}^{-1}_{CS,p})$ to them to conclude the proof.
- Assume that $bp \lesssim^p_{CS} bq$. If $I(p) = I(q)$ then we also have $p \lesssim^p_{CS} q$ and this corresponds to the situation above. However, in this case we could have $I(p) \subsetneq I(q)$; then $q = q' + r$, with r the summands $\sum_{a_i \in I(q) \setminus I(p)} a_i q_{ik}$, $I(p) = I(q')$, and $p \lesssim^p_{CS} q'$ and hence $p \sqsubseteq q'$. Now, we conclude the proof by applying the axiom $(\mathsf{S}_{CS,g})$ to q' and r. □

4 Axiomatization of the new simulation equivalences

Next we discuss the axiomatizability of the equivalences induced by covariant-contravariant and conformance simulations, obtaining a finite axiomatization for the latter, and also for the first, but only when the set A^{bi} of bivariant actions is empty. Instead, we also present the impossibility result proving that covariant-contravariant simulation is not axiomatizable if we have $A^{bi} \neq \emptyset$ and $A^r \cup A^l \neq \emptyset$.

4.1 Covariant-contravariant simulation

Let us first consider the case in which $A^{bi} = \emptyset$. In order to axiomatize the equivalence $\equiv_{CC}^{r,l}$ induced by (A^r, A^l)-simulation we apply the general procedure introduced in [9, 1, 8], based on the characterization

$$p \equiv_S p + q \Longleftrightarrow q \lesssim_S p \ .$$

Thus we obtain:

$(\mathsf{S1}_{\equiv}^{r,l})$ $\quad a_r(x + b_r y) = a_r(x + b_r y) + a_r x.$

$(\mathsf{S2}_{\equiv}^{r,l})$ $\quad a_r x = a_r x + a_r(x + b_l y).$

Obviously, the characterization above becomes unsound when contravariant prefixes appear because the pure contravariant simulation satisfies

$$q \equiv_S^{-1} p + q \Longleftrightarrow q \lesssim_S^{-1} p \ .$$

Therefore, we must reverse the inequalities above to obtain the adequate axioms for contravariant prefixes:

$(\mathsf{S3}_{\equiv}^{r,l})$ $\quad a_l x = a_l x + a_l(x + b_r y).$

$(\mathsf{S4}_{\equiv}^{r,l})$ $\quad a_l(x + b_l y) = a_l(x + b_l y) + a_l x.$

Now we would expect the set of axioms $\mathscr{A}_{CC}^{\equiv} = \{\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3, \mathsf{B}_4, \mathsf{S1}_{\equiv}^{r,l}, \mathsf{S2}_{\equiv}^{r,l}, \mathsf{S3}_{\equiv}^{r,l}, \mathsf{S4}_{\equiv}^{r,l}\}$ to axiomatize (A^r, A^l)-simulation equivalence. Certainly, all the axioms in this set are sound; in order to prove completeness in the absence of actions A^{bi}, we start by stating the following lemma that gives us two useful derived axioms.

Lemma 1 *The following equalities are derivable:*

$$\{\mathsf{S1}_{\equiv}^{r,l}, \mathsf{S2}_{\equiv}^{r,l}\} \vdash a_r(x + p_r) = a_r(x + p_r) + a_r(x + p_l) \qquad (\mathsf{DS1}_{\equiv}^{r,l})$$
$$\{\mathsf{S3}_{\equiv}^{r,l}, \mathsf{S4}_{\equiv}^{r,l}\} \vdash a_l(x + p_l) = a_l(x + p_l) + a_l(x + p_r) \qquad (\mathsf{DS2}_{\equiv}^{r,l})$$

where p_r (resp. p_l) denotes any process prefixed by actions in A^r (resp. A^l); more formally, $p_r = \sum_{i \in I} a_r^i p_i$ (resp. $p_l = \sum_{j \in J} a_l^j p_j$).

Proof. We only show the case of $(\mathsf{DS1}_{\equiv}^{r,l})$. We start by proving that $a_r(x + p_r) = a_r(x + p_r) + a_r x$ by induction over the size $|I|$ of I.

- If $|I| = 0$, the result is trivial.
- If $|I| = 1$, we immediately obtain the result by applying the axiom $(\mathsf{S1}_{\equiv}^{r,l})$.
- For $|I| > 1$, we take $I = I' \cup \{i\}$ with $|I'| = |I| - 1$. Note that $a_r(x + p_r) = a_r((x + p_r') + a_r^i p_i)$ so that, applying axiom $(\mathsf{S1}_{\equiv}^{r,l})$, we obtain

  $$a_r(x + p_r) = a_r((x + p_r') + a_r^i p_i) + a_r(x + p_r') = a_r(x + p_r) + a_r(x + p_r').$$

  Using the induction hypothesis with the term $a_r(x + p_r')$ leads to

  $$a_r(x + p_r) = a_r(x + p_r) + a_r(x + p_r') + a_r x,$$

  and, reusing the equality $a_r(x + p_r) + a_r(x + p_r') = a_r(x + p_r)$ above, we obtain

  $$a_r(x + p_r) = a_r(x + p_r) + a_r x \tag{1}$$

  as desired.

Now, we can analogously prove the equality

$$a_r x = a_r x + a_r(x + p_l). \tag{2}$$

Replacing $a_r x$ in equation 1 by the righthand side of equation 2 produces

$$a_r(x + p_r) = a_r(x + p_r) + a_r x + a_r(x + p_l)$$

and, applying equation 1 again, we finally obtain $(\mathsf{DS1}^{r,l}_{\equiv})$:

$$a_r(x + p_r) = a_r(x + p_r) + a_r(x + p_l).$$

□

For the main proof we have to adapt the classic technique for the completeness of the axiomatization of the plain simulation semantics ($p \lesssim_S q$ implies $\mathscr{A}_S \vdash q = p + q$), taking into account the difference between covariant and contravariant actions. For technical reasons we need to consider a "free" arbitrary term r.

Proposition 4 *If $p \lesssim_{CC} q$ then, for all processes r:*

$$\mathscr{A}^{\equiv}_{CC} \vdash a_r(q + r) = a_r(q + r) + a_r(p + r)$$

and

$$\mathscr{A}^{\equiv}_{CC} \vdash a_l(p + r) = a_l(p + r) + a_l(q + r).$$

Proof. We proceed by induction on the depth of p. We start by decomposing both p and q as follows: $p = p_r + p_l$, $q = q_r + q_l$, where $p_r = \sum_{i \in I_{p_r}} a^i_r p_i$, $p_l = \sum_{i \in I_{p_l}} a^i_l p_i$, $q_r = \sum_{i \in I_{q_r}} a^i_r q_i$ and $q_l = \sum_{i \in I_{q_l}} a^i_l q_i$. Then, it is clear that the depths of both p_r and p_l are less or equal than the depth of p and besides we have $p \lesssim_{CC} q \iff p_r \lesssim_{CC} q_r \wedge p_l \lesssim_{CC} q_l$.

Next, let us consider $p_r \lesssim_{CC} q_r$: this is an instance of the hypothesis of the statement to prove, which corresponds to the particular case in which $I(p) \cup I(q) \subseteq A_r$. Then, we need to prove both

$$\mathscr{A}^{\equiv}_{CC} \vdash a_r(q + r) = a_r(q + r) + a_r(p + r)$$

and

$$\mathscr{A}^{\equiv}_{CC} \vdash a_l(p + r) = a_l(p + r) + a_l(q + r).$$

Let us consider in detail the second statement.

- If $p = 0$, it follows that $\mathscr{A}^{\equiv}_{CC} \vdash a_l r = a_l r + a_l(q + r)$ by an application of the equation $(\mathsf{DS2}^{r,l}_{\equiv})$, with $p_l = 0$, $x = r$, and $p_r = q$.
- If $p = \sum_{i \in I} a^i_r p'_i$ and $q = \sum_{i \in J} a^i_r p'_i$, from $p \lesssim_{CC} q$ it follows, without loss of generality, that $I \subseteq J = I \cup J'$ and then we take $J = I \cup J'$ with J' chosen such that $J' \cap I = \emptyset$, with $p'_i \lesssim_{CC} q'_i$ for all $i \in I$. Now, by induction hypothesis, $\mathscr{A}^{\equiv}_{CC} \vdash a^i_r q'_i = a^i_r q'_i + a^i_r p'_i$. Next we obtain $\mathscr{A}^{\equiv}_{CC} \vdash \sum_{i \in I} a^i_r q'_i = \sum_{i \in I} a^i_r q'_i + p$ and hence, by adding $\sum_{i \in J'} a^i_r q'_i$ to both sides, $\mathscr{A}^{\equiv}_{CC} \vdash q = q + p$, by congruence, we have $\mathscr{A}^{\equiv}_{CC} \vdash q + r = q + p + r$. Now, by applying $(\mathsf{DS2}^{r,l}_{\equiv})$ with $x = p + r$, $p_l = 0$, and $p_r = q$, we obtain $\mathscr{A}^{\equiv}_{CC} \vdash a_l(p + r) = a_l(p + r) + a_l(p + r + q)$ which, combined with the previous equation, finally leads to $\mathscr{A}^{\equiv}_{CC} \vdash a_l(p + r) = a_l(p + r) + a_l(q + r)$.

The first statement above is proved in a similar way, and the ones arising from $p_l \lesssim q_l$ can be dealt with analogously.

To conclude, we consider the general case $p \lesssim_{CC} q$. By applying the results obtained above, starting from both $p_r \lesssim_{CC} q_r$ and $p_l \lesssim_{CC} q_l$, we have

$$\mathscr{A}_{CC}^{\equiv} \vdash a_r(q_r + r) = a_r(q_r + r) + a_r(p_r + r)$$

and

$$\mathscr{A}_{CC}^{\equiv} \vdash a_r(q_l + r) = a_r(q_l + r) + a_r(p_l + r).$$

In particular, making r equal to $q_l + r'$ in the first equality:

$$\mathscr{A}_{CC}^{\equiv} \vdash a_r(q_r + q_l + r') = a_r(q_r + q_l + r') + a_r(p_r + q_l + r').$$

(It is at this point that the "free" variable r in the statement is needed, so as to be able to proceed by instantiating it in a suitable manner). Now, instantiating r with $p_r + r'$ in the second derived equation:

$$\mathscr{A}_{CC}^{\equiv} \vdash a_r(p_r + q_l + r') = a_r(p_r + q_l + r') + a_r(p_r + p_l + r').$$

If we now combine the last two equations we can obtain

$$\mathscr{A}_{CC}^{\equiv} \vdash a_r(q_r + q_l + r') = a_r(q_r + q_l + r') + a_r(p_r + p_l + r'),$$

and, since r' is arbitrary, we finally get

$$\mathscr{A}_{CC}^{\equiv} \vdash a_r(q + r) = a_r(q + r) + a_r(p + r).$$

We can proceed in a similar way for a_l, thus obtaining

$$\mathscr{A}_{CC}^{\equiv} \vdash a_l(p + r) = a_l(p + r) + a_l(q + r).$$

And this concludes the proof. □

The main theorem is now at hand.

Theorem 1 *Whenever $A = A^r \cup A^l$, the set of axioms $\mathscr{A}_{CC}^{\equiv} = \{\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3, \mathsf{B}_4, \mathsf{S1}_{\equiv}^{r,l}, \mathsf{S2}_{\equiv}^{r,l}, \mathsf{S3}_{\equiv}^{r,l}, \mathsf{S4}_{\equiv}^{r,l}\}$ is complete for (A^r, A^l)-simulation equivalence.*

Proof. Let $p \equiv_{CC} q$: we need to prove $\mathscr{A}_{CC}^{\equiv} \vdash p = q$. The proof will follow by induction on the depth of p.

- If $p = 0$ we obviously have $q = 0$.
- Let $p = \sum_{i \in I} a_r^i p_r^i + \sum_{j \in J} a_l^j p_l^j$ and $q = \sum_{i \in I'} a_r^i q_r^i + \sum_{j \in J'} a_l^j q_l^j$. Then,
  - for each $i \in I$, there exists some $i' \in I'$ with $a_r^i = a_r^{i'}$ and $p_r^i \lesssim_{CC} q_r^{i'}$, and
  - for each $i' \in I'$ there exists some $i'' \in I$ with $a_r^{i'} = a_r^{i''}$ and $q_r^{i'} \lesssim_{CC} p_r^{i''}$.

  Obviously, it could be the case that $i \neq i''$. Then, we could repeat the same argument with $i_1 = i''$, and with $i_2 = i_1''$, ..., to obtain a sequence $(i, i_1, i_2, \ldots)$. Since $|I| < \infty$, eventually we will find $i_m = i_n$ and, hence,
  - for each $i \in I$ we obtain $i' \in I'$ and $i'' \in I$ such that $a_r^i = a_r^{i'} = a_r^{i''}$, $p_r^i \lesssim_{CC} q_r^{i'}$ and $p_r^{i''} \equiv_{CC} q_r^{i'}$.

Of course, we can repeat the same reasoning starting with $i' \in I'$ as well as for the contravariant summands in a dual way, to obtain the following decompositions:

$$p = \sum_{i\in I} a_r^i p_r^i + \sum_{k\in K} a_r^k p_r^k + \sum_{k'\in K'} a_l^{k'} p_l^{k'} + \sum_{m\in M} a_l^m p_l^m$$

and,

$$q = \sum_{i'\in I'} a_r^{i'} q_r^{i'} + \sum_{k\in K} a_r^k q_r^k + \sum_{k'\in K'} a_l^{k'} q_l^{k'} + \sum_{m'\in M'} a_l^{m'} q_l^{m'}$$

where:

- for all $i \in I$, there exists $k \in K$ such that $a_r^i = a_r^k$ and $p_r^i \lesssim_{CC} p_r^k$; and
- for all $m \in M$, there exists $k' \in K'$ such that $a_l^m = a_l^{k'}$ and $p_l^{k'} \lesssim_{CC} p_l^m$; and
- for all $i' \in I'$, there exists $k \in K$ such that $a_r^{i'} = a_r^k$ and $q_r^{i'} \lesssim_{CC} q_r^k$; and
- for all $m' \in M'$, there exists $k' \in K'$ such that $a_l^{m'} = a_l^{k'}$ and $q_l^{k'} \lesssim_{CC} q_l^m$; and
- for all $k \in K$, $p_r^k \equiv_{CC} q_r^k$; and
- for all $k' \in K'$, $p_l^{k'} \equiv_{CC} q_l^{k'}$.

Then we can apply the induction hypothesis to any pair (p_r^k, q_r^k) and also to any pair $(p_l^{k'}, q_l^{k'})$. To conclude the proof we only need to apply Proposition 4, taking $r = 0$, to any such pairs (p_r^i, p_r^k) and $(p_l^{k'}, p_l^m)$, and analogously for the components of q. □

The addition of bivariant actions (assuming that there are already other actions present) changes the picture completely. Now, it is no longer possible to axiomatize the equivalence.

Theorem 2 *If $A^{bi} \neq \emptyset$ and $A^r \cup A^l \neq \emptyset$, then (A^r, A^l)-simulation equivalence is not finitely axiomatizable.*

Proof. Let us take $a_{bi} \in A^{bi}$ and, without loss of generality, $a_r \in A^r$. We consider the two families of processes

$$p_n = a_r a_{bi}^n a_r 0 \quad \text{and} \quad q_n = a_r a_{bi}^n a_r 0 + a_r a_{bi}^n 0,$$

where, as usual, we denote by a_{bi}^n (with $n \geq 0$) the repeated application of the prefix operator a_{bi} (n times).

It is easy to check that $p_n \equiv_{CC} q_n$. On the one hand, $p_n \lesssim_{CC} q_n$ trivially; on the other hand, checking that $q_n \lesssim_{CC} p_n$ simply amounts to checking that $0 \lesssim_{CC} a_r$. (However, note that taking $p_n^- = a_{bi}^n a_r 0$ and $q_n^- = a_{bi}^n a_r 0 + a_{bi}^n 0$ does not lead to $p_n^- \equiv_{CC} q_n^-$; indeed, $p_n^- \not\lesssim_{CC} q_n^-$ because if we start with the first a_{bi} from the second summand of q_n^- then $a_{bi}^{n-1} a_r 0 \not\lesssim_{CC} a_{bi}^{n-1} 0$.) Now, for any finite axiomatization $\mathscr{A}$, let n be bigger than the depth of any term appearing in $\mathscr{A}$; we are going to show that if $\mathscr{A}$ is sound for $\equiv_{CC}$ then we cannot have $\mathscr{A} \vdash p_n = q_n$.

We will show that if we start with p_n and obtain a sequence of equivalent terms $p_n = p_n^1 = p_n^2 = \ldots$, where each term is obtained from the previous one by an application of a single axiom in $\mathscr{A}$, then no p_n^j can be q_n. If we apply an axiom to p_n in a position different from its root, then we are transforming a subprocess $p' = a_{bi}^m a_r 0$, with $m \leq n$, into some equivalent process $q \equiv_{CC} p'$. If we define $q \downarrow m$ as the process obtained by "pruning" q at depth m, the result will be bisimilar to $a_{bi}^m 0$, since q cannot execute any other action until it executes the prefix a_{bi} m times and, moreover, it cannot stop in the meantime. In a similar way, from $q \equiv_{CC} p'$ we also infer that $q \downarrow (m+1) \sim p' \downarrow (m+1)$ and then the obtained p_n^1 satisfies $p_n^1 \downarrow (n+2) \sim p_n$. The same argument can be applied starting from any p_n^j such that $p_n^j \downarrow (n+2) \sim p_n$, so that this invariant is preserved as long as there is no application of an axiom in $\mathscr{A}$ at the root of any p_n^j.

Therefore, the only possible way to break this invariant, that obviously is not satisfied by q_n, is to apply an axiom from $\mathscr{A}$ at the root of some p_n^j. In that case, the lefthand side of such an axiom would match several prefixes of the process $a_r a_{bi}^m 0$ and then, following [13], it is easy to see that the corresponding axiom has to be correct under bisimulation, too. As a consequence, the process p_{n+1}^j resulting after the application of the axiom also satisfies $p_{n+1}^j \downarrow (n+2) \sim p_n$. Therefore by repeated application of the axioms in $\mathscr{A}$ we will never reach a term such as q_n, thus concluding $\mathscr{A} \not\vdash p_n = q_n$. □

Note that the proof would remain valid even if we allowed conditional axioms whose conditions only observed the process locally, since the key fact in the proof above is that in order to generate the choice at q_n we need to "see from the top" that the two branches below, even if different from each other, can be joined to obtain a process equivalent to p_n. But the branches cannot be joined bottom up, in a step by step fashion, since $p_n^- \not\equiv_{CC} q_n^-$. Therefore, a conditional axiomatization whose conditions observe the processes locally would suffer the same problems as a purely equational one.

4.2 Conformance simulation

As before, we start by applying to the axioms characterizing $\lesssim_{CS}^p$ the general procedure presented in [9, 1, 8]. In this case we obtain the following two axioms:

$(\mathsf{S}_{\equiv}^{CS})$ $\quad I(p) \cap I(q) = \emptyset \Longrightarrow ap = ap + a(p+q).$

$(\mathsf{S}_{\equiv}^{-1,CS})$ $\quad I(q) \subseteq I(p) \Longrightarrow a(p+q) = a(p+q) + ap.$

Note that we have used the contravariant version of the procedure because once we compare two processes offering the same set of actions the behavior of $\lesssim_{CS}^p$ is contravariant since we have

$$ap \gtrsim_{CS}^p ap + aq.$$

Therefore, we cannot apply the general results in [9, 8] to prove the completeness of the proposed axiomatization. However, a beautiful variant of the classical proof for plain simulation will do the job.

Theorem 3 *The set of axioms $\mathscr{A}_{CS}^{\equiv} = \{\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3, \mathsf{B}_4, \mathsf{S}_{\equiv}^{CS}, \mathsf{S}_{\equiv}^{-1,CS}\}$ is a complete axiomatization for the simulation equivalence $\equiv_{CS}$.*

Proof. First note that $p \equiv_{CS} q$ implies $I(p) = I(q)$ and $p \equiv_{CS}^p q$, and therefore we can use either $\equiv_{CS}$ or $\equiv_{CS}^p$, indistinctly. It is also routine to check the correctness of the axioms for $\equiv_{CS}$. To prove completeness, we show that $p \lesssim_{CS}^p q$ implies $\mathscr{A}_{CS}^{\equiv} \vdash p = p + q$. Obviously, then we are done because $p \equiv_{CS} q$ implies $p \lesssim_{CS}^p q$ and $q \lesssim_{CS}^p p$.

We proceed by induction on the depth of p:

- $p = 0$ implies $q = 0$ trivially.
- Let $p \lesssim_{CS}^p q$ with $p \xrightarrow{a}$. Then we also have $q \xrightarrow{a}$ and for all q' with $q \xrightarrow{a} q'$ there exists $p \xrightarrow{a} p'$ such that $p' \lesssim_{CS} q'$. Note that we cannot conclude $p' \lesssim_{CS}^p q'$ since it is possible that $I(p') \subsetneq I(q')$, but then we can write $q' = q'' + r$ with $I(q'') = I(p')$ and $I(r) \cap I(q'') = \emptyset$. It is clear that $p' \lesssim_{CS}^p q''$, so that by induction hypothesis we obtain $\mathscr{A}_{CS}^{\equiv} \vdash p' = p' + q''$. Then, we have $\mathscr{A}_{CS}^{\equiv} \vdash ap' = a(p' + q'')$ and applying $(\mathsf{S}_{\equiv}^{-1,CS})$, $\mathscr{A}_{CS}^{\equiv} \vdash ap' = a(p' + q'') + aq''$, and then $\mathscr{A}_{CS}^{\equiv} \vdash ap' = ap' + aq''$. Now, by applying $(\mathsf{S}_{\equiv}^{CS})$ we have $\mathscr{A}_{CS}^{\equiv} \vdash aq'' = aq'' + a(q'' + r)$, to conclude that $\mathscr{A}_{CS}^{\equiv} \vdash ap' = ap' + aq'$ and therefore $\mathscr{A}_{CS}^{\equiv} \vdash p = p + q$. □

Note that $(\mathsf{S}^{-1,CS}_{\equiv})$ is the axiom characterizing the ready simulation equivalence, from which we conclude that $\equiv_{RS} \subseteq \equiv_{CS}$. Obviously, the reverse inclusion is false since $(\mathsf{S}^{CS}_{\equiv})$ is not sound for $\equiv_{RS}$. For instance, $ab =_{CS} ab + a(b+c)$, but $a(b+c) \not\lesssim_{RS} ab$. In fact, we also have $a(b+c) \not\lesssim_{S} ab$, proving that $\equiv_{CS} \not\subseteq \equiv_{S}$. In order to obtain $\equiv_{RS}$ from $\equiv_{CS}$ we should strengthen the definition of the latter by considering ready conformance simulations defined as plain conformance simulations, but only allowing pairs of processes satisfying $I(p) = I(q)$. If we denote by $\lesssim_{RCS}$ the generated preorder we have the following result.

Proposition 5 *$\lesssim_{RCS} = \lesssim^{-1}_{RS}$, and therefore $\equiv_{RCS} = \equiv_{RS}$ and $\lesssim^{-1}_{RS} \subseteq \lesssim_{CS}$.*

Since $(\mathsf{S}^{-1,CS}_{\equiv})$ is the axiom that defines ready simulation equivalence, it can be presented in an equivalent way avoiding the condition and thus obtaining a pure algebraic axiom. However, it is not clear whether axiom $(\mathsf{S}^{CS}_{\equiv})$ allows such a finite pure algebraic presentation, and in fact the same happens with the axiom (S_{CS}) in the axiomatization of the conformance preorder. Hence, it could be the case that both the conformance preorder and the induced equivalence are not finitely axiomatizable using pure equational axioms, as is the case for ready trace semantics.

5 Conclusions

We have continued with the study of covariant-contravariant simulation and conformance simulation semantics started in [12, 11] by considering the axiomatization of the preorders and equivalences that they define.

We have showed that the desired axiomatizations can be obtained from that of the plain simulation preorder, whose completeness proof can be adapted in a simple, but elegant manner to obtain the completeness of the new axiomatizations. Also, by applying a suitable variation of our "ready to preorder" techniques [9] we have obtained the axiomatizations of the corresponding conformance simulation equivalence. Surprisingly, we also succeeded in axiomatizating the equivalence for covariant-contravariant simulations but only in the particular case where $A^{bi} = \emptyset$; otherwise, we proved that the covariant-contravariant simulation equivalence has turned out to be the second known example of a semantics whose defining preorder can be finitely axiomatized, but the induced equivalence cannot. The first example of such a borderline situation can be found in [5]. It is curious to notice that although the two semantics are completely different (the semantics here is quite simple since it is a plain semantics, while the one in [5] is much more complicated), and in our case it is clear that the difficulties stem from the interference between bivariant and monovariant actions, the structure of the considered "counterexamples" in both cases is essentially the same: there is a choice betweeen two quite long branches which can be can joined into a single one, but this should be done in a single step because the choice cannot be delayed at all, even if the beginnings of the two branches are the same. Therefore, in order to capture the equivalence, we would need an axiom able to "see" the (too far away) ends of the two branches, but this is of course impossible with a finite number of axioms since the lengths of the branches in the counterexamples can be arbitrarily long.

We expect our work on the subject to contribute to a better understanding of all the complex situations that arise when covariant and contravariant concepts coexist. This, for example, is the case in all the recent works on modal, input-output or interface formalisms, that try to clarify the relationships betwen specifications and implementations. In fact, it is our intention to continue with this line of research by trying to discover, and take benefit from all the connections between our work and those cited in this paper.

References

[1] Luca Aceto, Wan Fokkink, and Anna Ingólfsdóttir. Ready to preorder: get your BCCSP axiomatization for free! In Till Mossakowski, Ugo Montanari, and Magne Haveraaen, editors, *Algebra and Coalgebra in Computer Science. Second International Conference, CALCO 2007, Bergen, Norway, August 20–24, 2007. Proceedings*, volume 4624 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2007.

[2] Adam Antonik, Michael Huth, Kim Larsen, Ulrik Nyman, and Andrzej Wasowski. 20 Years of Mixed and Modal Specifications. *Bulletin of the European Association for Theoretical Computer Science*, May 2008.

[3] Nikola Benes, Jan Kretínský, Kim Guldstrand Larsen, and Jirí Srba. On determinism in modal transition systems. *Theoretical Computer Science*, 410(41):4026–4043, 2009.

[4] Stephen D. Brookes and A. W. Roscoe. An improved failures model for communicating processes. In Stephen D. Brookes, A. W. Roscoe, and Glynn Winskel, editors, *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 281–305. Springer, 1984.

[5] Taolue Chen and Wan Fokkink. On the axiomatizability of impossible futures: preorder versus equivalence. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 156-165. 2008.

[6] Corina Cîrstea. A modular approach to defining and characterising notions of simulation. *Information and Computation*, 204(4):469–502, 2006.

[7] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *ESEC / SIGSOFT FSE*, pages 109–120, 2001.

[8] David de Frutos-Escrig, Carlos Gregorio-Rodríguez, and Miguel Palomino. On the unification of process semantics: Equational semantics. *Electronic Notes in Theoretical Computer Science*, 249:243–267, 2009.

[9] David de Frutos-Escrig, Carlos Gregorio-Rodríguez, and Miguel Palomino. Ready to preorder: an algebraic and general proof. *J. Log. Algebr. Program.*, 78(7):539–551, 2009.

[10] David de Frutos-Escrig, Fernando Rosa Velardo, and Carlos Gregorio-Rodríguez. New bisimulation semantics for distributed systems. In John Derrick and Jüri Vain, editors, *Formal Techniques for Networked and Distributed Systems — FORTE 2007, 27th IFIP WG 6.1 International Conference, Tallinn, Estonia, June 27-29, 2007, Proceedings*, volume 4574 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2007.

[11] Ignacio Fábregas, David de Frutos-Escrig, and Miguel Palomino. Logics for contravariant simulations. In John Hatcliff and Elena Zucca, editors, *FMOODS/FORTE 2010*, Lecture Notes in Computer Science. Springer. To appear.

[12] Ignacio Fábregas, David de Frutos-Escrig, and Miguel Palomino. Non-strongly stable orders also define interesting simulation relations. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *CALCO*, volume 5728 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2009.

[13] Jan Friso Groote. A new strategy for proving omega-completeness applied to process algebra. In Jos C. M. Baeten, and Jan Willem Klop, editors, *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 314–331. Springer, 1990.

[14] Jesse Hughes and Bart Jacobs. Simulations in coalgebra. *Theoretical Computer Science*, 327(1-2):71–108, 2004.

[15] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wasowski. Interface input/output automata. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *FM*, volume 4085 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2006.

[16] Kim Guldstrand Larsen and Bent Thomsen. A modal process logic. In *LICS*, pages 203–210. IEEE Computer Society, 1988.

[17] Guy Leduc. A framework based on implementation relations for implementing LOTOS specifications. *Computer Networks and ISDN Systems*, 25(1):23–41, 1992.

[18] Nancy Lynch. I/o automata: A model for discrete event systems. *In 22nd Annual Conference on Information Sciences and Systems*, pages 29–38, 1988.

[19] David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science, 5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981, Proceedings*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[20] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, and Roberto Passerone. Modal interfaces: unifying interface automata and modal specifications. In *EMSOFT '09: Proceedings of the seventh ACM international conference on Embedded software*, pages 87–96, New York, NY, USA, 2009. ACM.

[21] Jan Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.

Apéndice A

Versiones extendidas más relevantes

En este apéndice incluimos tres versiones extendidas de nuestras publicaciones. Las dos primeras son "Reflection and preservation of properties in coalgebraic (bi)simulations" y "Non-Strongly Stable Orders Also Define Interesting Simulation Relations", mientras que la tercera, titulada "On the specification of modal systems: a comparison of three frameworks", es la versión extendida y mejorada de nuestro artículo "Relating modal refinements, covariant-contravariant simulations and partial bisimulations" enviada recientemente a la revista "Science of Computer Programming" para su revisión.

Estas tres versiones extendidas incluyen y expanden tanto demostraciones como explicaciones extra. Además, como comentábamos en la página 12 estas versiones extendidas fueron nuestros trabajos de base, cuyas versiones publicadas se obtuvieron suprimiendo algunos resultados en función de las limitaciones concretas de las actas de los congresos.

Reflection and preservation of properties in coalgebraic (bi)simulations*

Ignacio Fábregas, Miguel Palomino, and David de Frutos Escrig

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

Abstract. Our objective is to extend the standard results of preservation and reflection of properties by bisimulations to the coalgebraic setting, as well as to study under what conditions these results hold for simulations. The notion of bisimulation is the classical one, while for simulations we use that proposed by Hughes and Jacobs. As for properties, we start by using a generalization of linear temporal logic to arbitrary coalgebras suggested by Jacobs, and then an extension by Kurtz which includes atomic propositions too.

1 Introduction

To reason about computational systems it is customary to mathematically formalize them by means of state-based structures such as labelled transitions systems or Kripke structures. This is a fruitful approach since it allows to study the properties of a system by relating it to some other, possibly better-known system, by means of simulations and bisimulations (see e.g., [14,13,11,3]).

The range of structures used to formalize computational systems is quite wide. In this context, coalgebras have emerged with a unifying aim [17]. A coalgebra is simply a function $c : X \longrightarrow FX$, where X is the set of states and FX is some expression on X (a functor) that describes the possible outcomes of a transition from a given state. Choosing different expressions for F one can obtain coalgebras that correspond to transition systems, Kripke structures, automata, ...

Coalgebras can also be related by means of (bi)simulations. Our goal in this paper is to prove that, like their concrete instantiations, (bi)simulations between arbitrary coalgebras preserve some interesting properties. A first step in this direction consists in choosing an appropriate notion for both bisimulation and simulation, as well as a logic in which to express these properties.

Bisimulations were originally introduced by Aczel and Mendler [1], who showed that the general definition coincided with the standard ones when particularized; it is an established notion. Simulations, on the other hand, were defined by Hughes and Jacobs [7] and lack such canonicity. Their notion of simulation depends on the use of orders that allow (perhaps too) much flexibility in what

* Research supported by the Spanish projects DESAFIOS TIN2006-15660-C02-01, WEST TIN2006-15578-C02-01 and PROMESAS S-0505/TIC/0407

it can be considered as a simulation; in order to show that simulations preserve properties, we will have to impose certain restrictions on such orders. As for the logic used for the properties, there is likewise no canonical choice at the moment. Jacobs proposes a temporal logic (see [8]) that generalizes linear temporal logic (LTL), though without atomic propositions; a clever insight of Pattinson [16] provides us with a way to endow Jacobs' logic with atomic propositions.

Since our original motivation was the generalization of the results about simulations and preservation of LTL properties, we will focus on Jacobs' logic and its extension with atomic propositions. Actually, modal logic seems to be the right logic to express properties of coalgebras and several proposals have been made in this direction, among them those in [9,12,16], which are invariant under behavioral equivalence. The reason for studying preservation/reflection of properties by bisimulations here is twofold: on the one hand, some of the operators in Jacobs' logic do not seem to fall under the framework of those general proposals; on the other hand, some of the ideas and insights developed for that study are needed when tackling simulations. As far as we know, reflection of properties by simulations in coalgebras has not been considered before in the literature.

2 Preliminaries

In this section we summarize definitions and concepts from [7,10,8], and introduce the notation we are going to use.

Given a category $\mathbb{C}$ and an endofunctor F in $\mathbb{C}$, an F-coalgebra, or just a coalgebra, consists of an object $X \in \mathbb{C}$ together with a morphism $c : X \longrightarrow FX$. We often call X the state space and c the transition or coalgebra structure.

Example 1. We show how two well-known structures can be seen as coalgebras:

- Labelled transition systems are coalgebras for the functor $F = \mathcal{P}(id)^A$, where A is the set of labels.
- Kripke structures are coalgebras for the functor $F = \mathcal{P}(AP) \times \mathcal{P}(id)$, where AP is a set of atomic propositions.

It is well-known that an arbitrary endofunctor F on **Sets** can be lifted to a functor in the category **Rel** of relations, that is, $\mathrm{Rel}(F) : \mathbf{Rel} \longrightarrow \mathbf{Rel}$. Given a relation $R \subseteq X \times Y$, its lifting is defined by

$$\mathrm{Rel}(F)(R) = \{\langle u, v\rangle \in FX_1 \times FX_2 \mid \exists w \in F(R).\, F(r_1)(w) = u, F(r_2)(w) = v\}\ ,$$

where $r_i : R \longrightarrow X_i$ are the projection morphisms.

A predicate P of a coalgebra $c : X \longrightarrow FX$ is just a subset of the state space. Also, a predicate $P \subseteq X$ can be lifted to a functor structure using the relation lifting:

$$\mathrm{Pred}(F)(P) = \coprod_{\pi_1}(\mathrm{Rel}(F)(\coprod_{\delta}(P))) = \coprod_{\pi_2}(\mathrm{Rel}(F)(\coprod_{\delta}(P))),$$

where $\delta = \langle id, id \rangle$ and $\coprod_f(X)$ is the image of X under f, so $\coprod_{\delta_x}(P) = \{(x,x) \mid x \in P\}$, $\coprod_{\pi_1}(R) = \{x_1 \mid \exists x_2 . x_1 R x_2\}$ is the domain of the relation R, and $\coprod_{\pi_2}(R) = \{x_2 \mid \exists x_1 . x_1 R x_2\}$ is its codomain.

The class of polynomial endofunctors is defined as the least class of endofunctors on **Sets** such that it contains the identity and constant functors, and is closed under product, coproduct, constant exponentiation, powerset and finite sequences. For polynomial endofunctors, $\mathrm{Rel}(F)$ and $\mathrm{Pred}(F)$ can be defined by induction on the structure of F. For further details on these definitions see [8]; we will introduce some of those when needed. For example, for the cases of labelled transition systems and Kripke structures we have:

$$\mathrm{Rel}(\mathcal{P}(id)^A)(R) = \{(f,g) \mid \forall a \in A.\, (f(a), g(a)) \in \{(U,V) \mid \forall u \in U.\, \exists v \in V.\, uRv \wedge \forall v \in V.\, \exists u \in U.\, uRv\}\}$$

$$\mathrm{Pred}(\mathcal{P}(id)^A)(P) = \{f \mid \forall a \in A.\, f(a) \in \{U \mid \forall u \in U.\, Pu\}\}$$

$$\mathrm{Rel}(\mathcal{P}(AP) \times \mathcal{P}(id))(R) = \{((u_1,u_2),(v_1,v_2)) \mid (u_1 = v_1.\, u_1, v_1 \in \mathcal{P}(AP)) \wedge (u_2,v_2) \in \{(U,V) \mid \forall u \in U.\, \exists v \in V.\, uRv \wedge \forall v \in V.\, \exists u \in U.\, uRv\}\}$$

$$\mathrm{Pred}(\mathcal{P}(AP) \times \mathcal{P}(id))(P) = \{(u,v) \mid (u \subseteq \mathcal{P}(AP)) \wedge (v \in \{U \mid \forall u \in U.\, Pu)\}$$

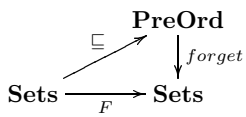
A bisimulation for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ which is "closed under c and d":

$$\text{if } (x,y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)(R)\,.$$

In the same way, an invariant for a coalgebra $c : X \longrightarrow FX$ is a predicate $P \subseteq X$ such that it is "closed under c", that is,

$$\text{if } x \in P \text{ then } c(x) \in \mathrm{Pred}(F)(P)\,.$$

We will use the definition of simulation introduced by Hughes and Jacobs in [7] which uses an order $\sqsubseteq$ for functors F that makes the following diagram commute



Given an order $\sqsubseteq$ on F, a simulation for the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ such that

$$\text{if } (x,y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)_{\sqsubseteq}(R)\,,$$

where $\mathrm{Rel}(F)_{\sqsubseteq}(R)$ is defined as

$$\mathrm{Rel}(F)_{\sqsubseteq}(R) = \{(u,v) \mid \exists w \in F(R).\ u \sqsubseteq Fr_1(w) \wedge Fr_2(w) \sqsubseteq v\}\,.$$

To express properties we will use a generalization of LTL proposed by Jacobs (see [8]) that applies to arbitrary coalgebras, whose formulas are given by the following BNF expression:

$$\varphi = P \subseteq X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi\,\mathcal{U}\,\varphi$$

$\bigcirc$ is the *nexttime* operator and its semantics (abusing notation) is defined as $\bigcirc P = c^{-1}(\mathrm{Pred}(F)(P)) = \{x \in X \mid c(x) \in \mathrm{Pred}(F)(P)\}$; $\Box$ is the *henceforth* operator defined as $\Box P = \exists Q \subseteq X.\ Q$ is an invariant for c, and $Q \subseteq P$ with $x \in Q$ or, equivalently by means of the greatest fixed point ν, $\Box P = \nu S.(P \wedge \bigcirc S)$; $\Diamond$ is the *eventually* operator defined as $\Diamond P = \neg\Box\neg P$; and $\mathcal{U}$ is the *until* operator defined as $P\,\mathcal{U}\,Q = \mu S.(Q \vee (P \wedge \neg \bigcirc \neg S))$, where μ is the least fixed point.

We denote the set of states in X that satisfies φ as $[\![\varphi]\!]_X$. That is, if $P \subseteq X$ is a predicate, then $[\![P]\!]_X = P$; if $\alpha \in \{\neg, \bigcirc, \Box, \Diamond\}$ then $[\![\alpha\varphi]\!]_X = \alpha[\![\varphi]\!]_X$, and if $\beta \in \{\wedge, \vee, \Rightarrow, \mathcal{U}\ \}$ then $[\![\varphi_1 \beta \varphi_2]\!]_X = [\![\varphi_1]\!]_X \beta [\![\varphi_2]\!]_X$. We will usually omit the reference to the set X when it is clear from the context. We say that an element x satisfies a formula φ, and we denote it by $c, x \models \varphi$, when $x \in [\![\varphi]\!]$. Again, we will usually omit the reference to the coalgebra c.

3 Reflection and preservation in bisimulations

These definitions of reflection and preservation are slightly more involved than for classical LTL because the logic proposed by Jacobs does not use atomic propositions, but predicates (subsets of the set of states). Later, we will see how atomic propositions can be introduced in the logic.

Given a predicate P on X and a binary relation $R \subseteq X \times Y$, we will say that an element $y \in Y$ is in the direct image of P, and we will denote it by $y \in RP$, if there exists $x \in X$ with $x \in P$ and xRy. The inverse image of R is just the direct image for the relation R^{-1}.

Definition 1. *Given two formulas φ on X and ψ on Y, built over predicates $P_1, \ldots P_n$ and $Q_1, \ldots Q_n$, respectively, and a binary relation $R \subseteq X \times Y$, we define the image of φ as a formula φ^* on Y, obtained by substituting in φ RP_i for P_i. Likewise, we build ψ^{-1}, the inverse of ψ, substituting $R^{-1}Q_i$ for Q_i in ψ.*

Remark 1. It is important to notice that φ^* coincides with φ^{-1} when we consider R^{-1} instead of R. Analogously, φ^{-1} is just φ^* when we consider R^{-1} instead of R.

Now we can define when a relation preserves or reflects properties.

Definition 2. *Let $R \subseteq X \times Y$ be a binary relation and a and b elements such that aRb. We say that R* preserves *the property φ on X if, whenever $a \models \varphi$, $b \models \varphi^*$. We say that R* reflects *the property φ on Y if $b \models \varphi$ implies $a \models \varphi^{-1}$.*

For the sake of the clarity of the proofs we will present the results on preservation and reflection of properties with one proposition for each temporal operator instead of one main theorem.

Let us first prove a couple of technical lemmas.

Lemma 1. *Let F be a polynomial functor, $R \subseteq X \times Y$ a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, $P \subseteq Y$ and xRy. If $d(y) \in \mathrm{Pred}(F)(P)$, then $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.*

Proof. We are going to prove this result by structural induction on the functor F.

1. $F = cnst$. In this case $\mathrm{Pred}(F)(P) = \top$ and also $\mathrm{Pred}(F)(R^{-1}P) = \top$, and $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$ trivially.
2. $F = id$. In this case $\mathrm{Pred}(F)(P) = P$ for all P and we have to prove that $c(x) \in R^{-1}P$, that is, that there exists some $b \in P$ with $c(x)Rb$. If we take $b = d(y)$, then we have the result because R is a bisimulation.
3. $F = F_1 \times F_2$. In this case we have

$$\mathrm{Pred}(F)(P) = \{(u, v) \mid \mathrm{Pred}(F_1)(P)(u) \wedge \mathrm{Pred}(F_2)(P)(v)\}\,.$$

   Let us suppose that $d(y) = (d_1(y), d_2(y))$ and $c(x) = (c_1(x), c_2(x))$. Therefore, if $d(y) \in \mathrm{Pred}(F)(P)$ then we also have $d_1(y) \in \mathrm{Pred}(F_1)(P)$ and $d_2(y) \in \mathrm{Pred}(F_2)(P)$. However, since R is a bisimulation between c and d and xRy, we have $c(x)\mathrm{Rel}(F)(R)d(y)$, where

$$\mathrm{Rel}(F)(R) = \{((u_1, u_2), (v_1, v_2)) \mid \mathrm{Rel}(F_1)(R)(u_1, v_1) \wedge \mathrm{Rel}(F_2)(R)(u_2, v_2)\}\,.$$

   So, in particular, we have

$$c_1(x)\mathrm{Rel}(F_1)(R)d_1(y)$$

   as well as

$$c_2(x)\mathrm{Rel}(F_2)(R)d_2(y)\,.$$

   That is, R is also a bisimulation for c_1 and d_1 and c_2 and d_2. Now we can apply our induction hypothesis and since $d_1(y) \in \mathrm{Pred}(F_1)(P)$, we get $c_1(x) \in \mathrm{Pred}(F_1)(R^{-1}P)$ and, analogously $c_2(x) \in \mathrm{Pred}(F_2)(R^{-1}P)$, so $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$, as we wanted to prove.
4. $F = F_1 + F_2$. In this case we have

$$\mathrm{Pred}(F)(P) = \{\kappa_1(u) \mid \mathrm{Pred}(F_1)(P)(u)\} \cup \{\kappa_2(v) \mid \mathrm{Pred}(F_2)(P)(v)\}\,.$$

   Let us suppose that $d(y) = \kappa_1(d_1(y)) = (d_1(y), 1)$; we must have that $d_1(y) \in \mathrm{Pred}(F_1)(P)$. Let us consider the constant coalgebras:

$$\begin{array}{rl} c_X : X \to F_1X & \qquad d_Y : Y \to F_1Y \\ z \mapsto c_1(x) & \qquad z \mapsto d_1(y) \end{array}$$

Trivially, R is a bisimulation between c_X and d_Y; then, if we apply the induction hypothesis we get $c_1(x) \in \mathrm{Pred}(F_1)(R^{-1}P)$ and hence $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$. Reasoning in an analogous way we get that if $d(y) = \kappa_2(d_1(y))$, also $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.

5. $F = G^A$. In this case,

$$\mathrm{Pred}(F)(P) = \{f \mid \forall a \in A.\, \mathrm{Pred}(G)(P)(f(a))\}\,.$$

Now, for each $a \in A$ and F-coalgebra $d : Y \longrightarrow F(Y)$ we can define a coalgebra in G: $d^a : Y \longrightarrow G(Y)$ where, for each $y \in Y$, $d^a(y) = d(y)(a)$; analogously we define $c^a(x) = c(x)(a)$ for all $x \in X$. In this way we have that xRy and $d^a(y) = d(y)(a) \in \mathrm{Pred}(G)(P)$. Now, using the definition of the relation lifting for the exponent functor and the fact that R is a bisimulation between c and d it follows that R is also a bisimulation between c^a and d^a. Applying the induction hypothesis we get $c^a(x) \in \mathrm{Pred}(G)(R^{-1}P)$. Since this argument is valid for all $a \in A$, we get $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$, as we wanted to prove.

6. $F = \mathcal{P}(G)$. In this case

$$\mathrm{Pred}(F)(P) = \{U \mid \forall u \in U.\, \mathrm{Pred}(G)(P)(u)\}\,.$$

We have $d(y) \in \mathrm{Pred}(F)(P)$, so for all $b \in d(y)$ it is true that $b \in \mathrm{Pred}(G)(P)$ and we want to prove that $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$ or equivalently that for all $a \in c(x)$ we have $a \in \mathrm{Pred}(G)(R^{-1}P)$. Let us take $a \in c(x)$ and define a constant coalgebra:

$$\begin{aligned} c^a_X : X &\to GX \\ z &\mapsto a \end{aligned}$$

Now, from our hypothesis that xRy and R is a bisimulation, by definition we have $c(x)\mathrm{Rel}(F)(R)d(y)$. By the definition of relation lifting it follows that there exists some $b \in d(y)$ such that $a\mathrm{Rel}(G)(R)b$.
Now we consider another constant coalgebra

$$\begin{aligned} d^b_Y : Y &\to GY \\ z &\mapsto b \end{aligned}$$

Trivially, R is a bisimulation between the coalgebras c^a_X and d^b_Y because $a\mathrm{Rel}(G)(R)b$; also $d^b_Y = b \in \mathrm{Pred}(G)(P)$, so by induction hypothesis it follows that $c^a_X = a \in \mathrm{Pred}(G)(R^{-1}P)$. This argument is valid for all $a \in c(x)$, therefore, as we wanted to prove, $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.

7. $F = G^*$. In this case,

$$\mathrm{Pred}(F)(P) = \{\langle u_1, \ldots, u_n\rangle \mid \forall i \leq n.\, \mathrm{Pred}(G)(P)(u_i)\}\,.$$

Let us suppose that $d(y) = \langle d_1(y), \ldots, d_n(y)\rangle$ and $c(x) = \langle c_1(x), \ldots, c_n(x)\rangle$. Then, for each $i \leq n$ we have $d_i(y) \in \mathrm{Pred}(G)(P)$. Now, we define these two families of constant coalgebras:

$$\begin{aligned} c^x_i : X &\to GX \\ z &\mapsto c_i(x) \end{aligned} \qquad\qquad \begin{aligned} d^y_i : Y &\to GY \\ z &\mapsto d_i(y) \end{aligned}$$

Trivially, R is a bisimulation between c_i^x and d_i^y for each $i \leq n$. Therefore, using the induction hypothesis we get $c_i(x) \in \mathrm{Pred}(G)(R^{-1}P)$ for each $i \leq n$, that is, $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.

□

As a direct consequence of this lemma and the fact that the inverse of a bisimulation, is still a bisimulation we also have the following result.

Lemma 2. *Let F be a polynomial functor, $R \subseteq X \times Y$ a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, $P \subseteq X$ and xRy. If $c(x) \in \mathrm{Pred}(F)(P)$, then $d(y) \in \mathrm{Pred}(F)(RP)$.*

Another auxiliary lemma we need to prove the main result of this section is the following:

Lemma 3. *The direct and inverse images of an invariant are also invariants.*

Proof. Let R be a bisimulation between $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Let us suppose that $P \subseteq X$ is an invariant and let us prove that RP is too; that is, for all $y \in RP$ it must be the case that $d(y) \in \mathrm{Pred}(F)(RP)$. If $y \in RP$, then there exists $x \in P$ such that xRy. Since P is an invariant, we also have $c(x) \in \mathrm{Pred}(F)(P)$ and by Lemma 1 we get $d(y) \in \mathrm{Pred}(F)(RP)$.

On the other hand, since R^{-1} is also a bisimulation, the inverse image of an invariant is an invariant too.

At this point it is interesting to recall that our objective is to prove that bisimulations preserve and reflect properties of a temporal logic, that is, if we have xRy and $y \models \varphi$ then we must also have $x \models \varphi^{-1}$; and, analogously, if $x \models \varphi$ then $y \models \varphi^*$. We will show this result for all temporal operators except for the negation; it is well-known that negation is reflected and preserved by standard bisimulations, but not here because of the lack of atomic propositions in the coalgebraic temporal logic.

To prove the result for the rest of temporal operators, we will see that if $y \in [\![\varphi]\!]$ then we also have $x \in R^{-1}[\![\varphi]\!]$ and, analogously, if $x \in [\![\varphi]\!]$ then $y \in R[\![\varphi]\!]$. Ideally, we would like to have both $R^{-1}[\![\varphi]\!] = [\![\varphi^{-1}]\!]$ and $R[\![\varphi]\!] = [\![\varphi^*]\!]$ but, in general, only the inclusion $\subseteq$ is true. Fortunately this is enough to prove our propositions, since the temporal operators are all monotonic except for the negation. In fact, here is where the problem with negation appears.

Before continuing, it is interesting to show that the other inclusion is not true.

Let us take $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2\}$, the functor $F = id$ and the coalgebras $c : X \longrightarrow X$ and $d : Y \longrightarrow Y$ defined as $c(x_1) = x_2$, $c(x_2) = x_3$, $c(x_3) = x_3$, $d(y_1) = y_2$ and $d(y_2) = y_2$. Then, we take R as the bisimulation between c and d defined as $R = \{(x_3, y_1), (x_3, y_2), (x_1, y_2), (x_2, y_2), (x_3, y_2)\}$. We also consider the predicate $P = \{y_1\}$ and the formula $\varphi = \bigcirc\varphi_0$, where $\varphi_0 = \bigcirc P$. Let us show that $x_1 \in [\![\varphi^{-1}]\!]$ but $x_1 \notin R^{-1}[\![\varphi]\!]$:

– $x_1 \in [\![\varphi^{-1}]\!]$. By definition of inverse of a formula we have $\varphi^{-1} = \bigcirc\bigcirc R^{-1}P$, who also, by definition of R, is equal to $\bigcirc\bigcirc\{x_3\}$. Therefore, $x_1 \in [\![\varphi^{-1}]\!]$ if and only if $x_1 \in \bigcirc\bigcirc\{x_1\}$, in other words, if $c(x_1) = x_2 \in \mathrm{Pred}(F)(\bigcirc\{x_3\}) = \bigcirc\{x_3\}$ that, in turn, is equivalent to $c(x_2) = x_3 \in \mathrm{Pred}(F)(\{x_3\}) = \{x_3\}$. That way we conclude that $x_1 \in [\![\varphi^{-1}]\!]$.
– $x_1 \notin R^{-1}[\![\varphi]\!]$. Let us suppose that it is not true, in other words, $x_1 \in R^{-1}[\![\varphi]\!]$. Then, by definition of inverse predicate it would exist some $y \in Y$ such that $x_1 R y$ with $y \in [\![\varphi]\!]$. By definition of R the only possible candidate is y_2. So we get $y_2 \in [\![\varphi]\!] = \bigcirc\bigcirc\{y_1\}$, which is equivalent to $d(y_2) = y_2 \in \mathrm{Pred}(F)(\bigcirc\{y_1\}) = \bigcirc\{y_1\}$, that is, $d(y_2) = y_2 \in \mathrm{Pred}(F)(\{y_1\}) = \{y_1\}$. But the latter is a contradiction, so we have proved that $x_1 \notin R^{-1}[\![\varphi]\!]$.

he proof of the next is not specially difficult except for the case of the *until* operator, that will need an auxiliary result that we state now but will be proved later.

Lemma 4. *Let φ_1, φ_2 be temporal formulas which do not contain the negation operator and R a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ such that xRy. If $y \in [\![\varphi_1]\!]\,\mathcal{U}\,[\![\varphi_2]\!]$, then $x \in R^{-1}[\![\varphi_1]\!]\,\mathcal{U}\,R^{-1}[\![\varphi_2]\!]$.*

Lemma 5. *Let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. For all temporal formulas φ which do not contain the negation operator, it follows that*

$$R^{-1}[\![\varphi]\!]_Y \subseteq [\![\varphi^{-1}]\!]_X\,.$$

Proof. We will proceed by structural induction on φ.

1. $\varphi = P$, where $P \subset Y$ is a predicate. In this case $R^{-1}[\![\varphi]\!] = R^{-1}P = [\![\varphi^{-1}]\!]$, by definition.
2. $\varphi = \varphi_1 \vee \varphi_2$, or $\varphi = \varphi_1 \wedge \varphi_2$. In both cases we trivially get the result.
3. $\varphi = \bigcirc\varphi_1$. Let us suppose $x \in R^{-1}[\![\bigcirc\varphi_1]\!]$; then it exists y such that xRy with $y \in [\![\bigcirc\varphi_1]\!] = \bigcirc[\![\varphi_1]\!]$. Equivalently, we have xRy with $d(y) \in \mathrm{Pred}(F)([\![\varphi_1]\!])$ so by Lemma 2 we have $c(x) \in \mathrm{Pred}(F)(R^{-1}[\![\varphi_1]\!])$. Now, by induction hypothesis we know that $R^{-1}[\![\varphi_1]\!] \subseteq [\![\varphi_1^{-1}]\!]$ and by monotony of operator $\mathrm{Pred}(F)$ we obtain $c(x) \in \mathrm{Pred}(F)([\![\varphi_1^{-1}]\!])$, that is, $x \in [\![\varphi^{-1}]\!]$.
4. $\varphi = \Box\varphi_1$. By definition $\Box[\![\varphi_1]\!]$ is the greatest invariant contained in $[\![\varphi_1]\!]$ and, henceforth by Lemma 3, $R^{-1}\Box[\![\varphi_1]\!]$ is also an invariant. Also, as $\Box[\![\varphi_1]\!] \subseteq [\![\varphi_1]\!]$, is $R^{-1}\Box[\![\varphi_1]\!] \subseteq R^{-1}[\![\varphi_1]\!]$. By induction hypothesis $R^{-1}[\![\varphi_1]\!] \subseteq [\![\varphi_1^{-1}]\!]$, so $R^{-1}\Box[\![\varphi_1]\!]$ is an invariant contained in $[\![\varphi_1^{-1}]\!]$ and thus contained in $\Box[\![\varphi_1^{-1}]\!]$, as we wanted to prove.
5. $\varphi = \Diamond\varphi_1$. Let us suppose $x \in R^{-1}\Diamond[\![\varphi_1]\!]$; then it exists some y such that xRy with $y \in \Diamond[\![\varphi_1]\!]$, that is, $y \in \neg\Box\neg[\![\varphi_1]\!]$. We recall that we must prove $x \in \neg\Box\neg[\![\varphi_1^{-1}]\!]$, assuming xRy and $y \in \neg\Box\neg[\![\varphi_1]\!]$; using the counter-reciprocal, we will see that xRy and $x \in \Box\neg[\![\varphi_1^{-1}]\!]$ implies $y \in \Box\neg[\![\varphi_1]\!]$.
   Let us take $x \in \Box\neg[\![\varphi_1^{-1}]\!]$. By definition, we know that it must exist an invariant S such that $S \subseteq \neg[\![\varphi_1^{-1}]\!]$ with $x \in S$. Let us see that RS is an invariant such that $RS \subseteq \neg[\![\varphi_1]\!]$ with $y \in RS$ (if it is so, then we will get

$y \in \Box\neg[\![\varphi_1]\!]$). Indeed, since S is an invariant, then RS is also an invariant and, trivially, $y \in RS$. Let us see now that $RS \subseteq \neg[\![\varphi_1]\!]$. To prove this, let us suppose that it is false, that is, there exists an element $b \in RS$ such that $b \in [\![\varphi_1]\!]$. But since $b \in RS$, then it must exist $a \in S \subseteq \neg[\![\varphi_1^{-1}]\!]$, that is, $a \notin [\![\varphi_1^{-1}]\!]$. On the other hand, since $b \in [\![\varphi_1]\!]$ and aRb then $a \in R^{-1}[\![\varphi_1]\!]$, that by induction hypothesis is contained in $[\![\varphi_1^{-1}]\!]$, that is, $a \in [\![\varphi_1^{-1}]\!]$, but this is a contradiction so $RS \subseteq \neg[\![\varphi_1]\!]$.

6. $\varphi = \varphi_1 \ \mathcal{U} \ \varphi_2$. Let us suppose that $x \in R^{-1}[\![\varphi_1 \ \mathcal{U} \ \varphi_2]\!]$, then there exists y such that xRy with $y \in [\![\varphi_1 \ \mathcal{U} \ \varphi_2]\!] = [\![\varphi_1]\!] \ \mathcal{U} \ [\![\varphi_2]\!]$. By Lemma 4 we get $x \in R^{-1}[\![\varphi_1]\!] \ \mathcal{U} \ R^{-1}[\![\varphi_2]\!]$ that is, for all S we must have $x \in R^{-1}[\![\varphi_2]\!] \cup (R^{-1}[\![\varphi_1]\!] \cap \neg \bigcirc \neg S)$. By induction hypothesis, we have $R^{-1}[\![\varphi_i]\!] \subseteq [\![\varphi_i^{-1}]\!]$ for $i = 1, 2$, so, for all S we have $x \in [\![\varphi_2^{-1}]\!] \cup ([\![\varphi_1^{-1}]\!] \cap \neg \bigcirc \neg S)$, that is $x \in [\![\varphi_1^{-1}]\!] \ \mathcal{U} \ [\![\varphi_2^{-1}]\!]$.

□

Once again, due to the fact that R^{-1} is a bisimulation we also have:

Lemma 6. *Let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. For all temporal formulas φ which do not contain the negation operator, it follows that*

$$R[\![\varphi]\!]_X \subseteq [\![\varphi^*]\!]_Y \, .$$

Finally we can show that bisimulations reflect and preserve properties given by any temporal operator except for the negation.

The firsts two cases correspond to the reflection of the predicates and the reflection of formulas with elemental operators, whose proofs are trivial.

Proposition 1. *Let ψ be a predicate $P \subseteq Y$ and R a bisimulation between the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then the property ψ is reflected by R.*

Proposition 2. *Let $\psi = \varphi_1 \vee \varphi_2$ or $\psi = \varphi_1 \wedge \varphi_2$ where φ_1 and φ_2 are formulas reflected by bisimulations, and let R be a bisimulation between the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then the property ψ is reflected by R.*

Now we can show more interesting cases, like the reflection with the *nexttime* operator.

Proposition 3. *Let $\psi = \bigcirc\varphi$ be a formula on a set Y such that φ is reflected by bisimulations, and let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then the property ψ is reflected by R.*

Proof. Let aRb and suppose that $b \models \psi$, that is, $b \in \bigcirc[\![\varphi]\!]$ which, by definition, means that $d(b) \in \mathrm{Pred}(F)([\![\varphi]\!])$. We are going to show that $a \in \bigcirc R^{-1}[\![\varphi]\!]$ or, equivalently, $c(a) \in \mathrm{Pred}(F)(R^{-1}[\![\varphi]\!])$. But the latter is straightforward from Lemma 2.

Now, from Lemma 5 and the monotony of the nexttime operator we get $a \in \bigcirc[\![\varphi^{-1}]\!]$. □

The same result is true for the *henceforth* operator:

Proposition 4. *Let $\psi = \Box\varphi$ be a formula on Y such that φ is reflected by bisimulations and let R be a bisimulation between the coalgebras $c : X \longrightarrow F(X)$, $d : Y \longrightarrow F(Y)$. Then, the property ψ is reflected by R.*

Proof. Let aRb and let's suppose that $b \models \Box\varphi$, that is, there exists $S \subseteq Y$ such that S is an invariant, $S \subseteq [\![\varphi]\!]$ with $b \in S$. Once again, we will show that $a \in \Box R^{-1}[\![\varphi]\!]$ and then use Lemma 5 to get $a \in \Box[\![\varphi^{-1}]\!]$.

In fact, we only need to show that there exists an invariant $T \subseteq X$ such that $T \subseteq R^{-1}[\![\varphi]\!]$ and $a \in T$. If we take the invariant $T = R^{-1}S$, then $a \in T$ since aRb and $b \in S$; also $R^{-1}S \subseteq R^{-1}[\![\varphi]\!]$ because $S \subseteq [\![\varphi]\!]$. □

Now we can state and prove the corresponding proposition for the operator *eventually*.

Proposition 5. *Let $\psi = \Diamond\varphi$ be a formula on a set Y such that φ is reflected by bisimulations and let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then the property ψ is reflected by R.*

Proof. Let aRb and let us suppose $b \models \psi$; by definition, $b \in \neg\Box\neg[\![\varphi]\!]$. Once again, it will be enough to prove that $a \in \neg\Box\neg R^{-1}[\![\varphi]\!]$. So, let us assume that $a \in \Box\neg R^{-1}[\![\varphi]\!]$ and let us show that $b \in \Box\neg[\![\varphi]\!]$. Indeed, if $a \in \Box\neg R^{-1}[\![\varphi]\!]$ then there exists an invariant $S \subseteq X$ such that $S \subseteq \neg R^{-1}[\![\varphi]\!]$, with $a \in S$. But RS is an invariant such that $RS \subseteq \neg[\![\varphi]\!]$ and $b \in RS$, so we have proved the proposition. □

The proof of the corresponding proposition involving the *until* operator is a bit more difficult because it needs to resort to some more technical lemmas and auxiliary results. One of those will be Lemma 4, that, also, will be the base for proving the main proposition. To prove it we will need an auxiliary definition which will simplify the notation:

Definition 3. *Given a set Y and $P, Q \subseteq Y$, we define an operator $f^U_{(P,Q)}$ as:*

$$\begin{array}{rcl} f^U_{(P,Q)} : \mathcal{P}(Y) & \longrightarrow & \mathcal{P}(Y) \\ S & \longmapsto & Q \cup (P \cap \neg \bigcirc \neg S)\,. \end{array}$$

Notice that $y \in [\![\varphi_1]\!]\,\mathcal{U}\,[\![\varphi_2]\!]$ by definition is $y \in \mu S.([\![\varphi_2]\!] \vee ([\![\varphi_1]\!] \wedge \neg \bigcirc \neg S))$ that is,

$$y \in \mu S. f^U_{([\![\varphi_1]\!],[\![\varphi_2]\!])}(S)\,.$$

And, on the other hand, $x \in R^{-1}[\![\varphi_1]\!]\,\mathcal{U}\,R^{-1}[\![\varphi_2]\!]$ is equivalent to

$$x \in \mu S. f^U_{(R^{-1}[\![\varphi_1]\!],R^{-1}[\![\varphi_2]\!])}(S)\,.$$

We will also need two classical results, the first one is a definition of $\cup$-continuity (see for example [3]) and the second one says how to calculate least fixed points:

Definition 4. *Given a set S and a function $f : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$, we say that f is $\cup$-continuous if given an ascending chain of predicates $\{P_i \mid i \in \mathbb{N}\}$ with $P_i \subseteq P_{i+1}$ for each i, we have*

$$f(\bigcup_i P_i) = \bigcup_i f(P_i)\,.$$

Proposition 6. *If $f : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$ is monotonic and $\cup$-continuous, then*

$$\mu S.f(S) = \bigcup_{i=1}^{\infty} f^i(\emptyset)\,.$$

It is not hard to prove that the function $f^U_{(P,Q)}$ is both monotonic and $\cup$-continuous. First we shall see monotony: Let $S_1 \subseteq S_2$ and we will see $f^U_{(P,Q)}(S_1) \subseteq f^U_{(P,Q)}(S_2)$. Since $S_1 \subseteq S_2$ then $\neg S_2 \subseteq \neg S_1$ which, by monotony of $\bigcirc$, implies $\bigcirc\neg S_2 \subseteq \bigcirc\neg S_1$. That is, $\neg \bigcirc \neg S_1 \subseteq \neg \bigcirc \neg S_2$; by definition of $f^U_{(P,Q)}$, that is enough to show that $f^U_{(P,Q)}(S_1) \subseteq f^U_{(P,Q)}(S_2)$.

On the other hand we have to prove that $f^U_{(P,Q)}$ satisfies the following equation

$$f^U_{(P,Q)}(\bigcup_i P_i) = \bigcup_i f^U_{(P,Q)}(P_i)\,. \tag{1}$$

Unfolding in both sides of (1) we get

$$Q \cup (P \cap \neg \bigcirc (\bigcap_i \neg P_i)) = Q \cup (P \cap (\bigcup_i \neg \bigcirc \neg P_i))\,,$$

so it is enough to prove

$$\neg \bigcirc (\bigcap_i \neg P_i) = \bigcup_i \neg \bigcirc \neg P_i\,.$$

This last equation is equivalent to

$$\bigcirc(\bigcap_i \neg P_i) = \bigcap_i \bigcirc\neg P_i\,,$$

that is,

$$\mathrm{Pred}(F)(\bigcap_i \neg P_i) = \bigcap_i \mathrm{Pred}(F)(\neg P_i)\,. \tag{2}$$

And this last equation trivially follows from the fact that $\mathrm{Pred}(F)$ preserves intersections as it is shown in [8].

Before proving Lemma 4 it will be necessary to prove another technical lemma:

Lemma 7. *Given the function $f^U_{(P,Q)} : \mathcal{P}(Y) \longrightarrow \mathcal{P}(Y)$, the following is true*

$$R^{-1} f^U_{([\![\varphi_1]\!],[\![\varphi_2]\!])}(S) \subseteq f^U_{(R^{-1}[\![\varphi_1]\!],R^{-1}[\![\varphi_2]\!])}(R^{-1}S)\,.$$

Proof. We introduce now the following simplified notation:

$$f_1 \quad \text{denotes} \quad f^U_{([\![\varphi_1]\!],[\![\varphi_2]\!])}(S)\,,$$

$$f_2 \quad \text{denotes} \quad f^U_{(R^{-1}[\![\varphi_1]\!],R^{-1}[\![\varphi_2]\!])}(S)\,.$$

So, we have:

$$R^{-1}f_1(S) = R^{-1}[\![\varphi_2]\!] \cup (R^{-1}[\![\varphi_1]\!] \cap R^{-1}(\neg \bigcirc \neg S)))\,,$$

$$f_2(R^{-1}S) = R^{-1}[\![\varphi_2]\!] \cup (R^{-1}[\![\varphi_1]\!] \cap \neg \bigcirc \neg(R^{-1}S)))\,.$$

Indeed it will be enough to prove that $R^{-1}(\neg \bigcirc \neg S) \subseteq \neg \bigcirc \neg(R^{-1}S)$. So, if $x \in R^{-1}(\neg \bigcirc \neg S)$ then there exists some y such that xRy with $y \in \neg \bigcirc \neg S$; and we must prove that $x \in \neg \bigcirc \neg(R^{-1}S)$. Or, equivalently we can prove that given xRy, if $x \in \bigcirc\neg(R^{-1}S)$ then $y \in \bigcirc\neg S$.

We will prove this last result, by estructural induction on the functor F in the same way we did in Lemma 2.

1. $F = cnst$. In this case both $\mathrm{Pred}(F)(\neg R^{-1}S) = \top$ and $\mathrm{Pred}(F)(\neg S) = \top$, so we get the result.
2. $F = id$. In this case we have to show that if $c(x) \in \neg R^{-1}S$ then $d(y) \in \neg S$. Let us suppose that $d(y) \in S$. Then, since xRy, R is a bisimulation and we are working with the identity functor, we also have $c(x)Rd(y)$, and that way $c(x) \in R^{-1}S$; this contradicts the hypothesis, so it must be $d(y) \in \neg S$.
3. $F = F_1 \times F_2$. In this case we have $c_1(x) \in \mathrm{Pred}(F_1)(\neg R^{-1}S)$ and $c_2(x) \in \mathrm{Pred}(F_2)(\neg R^{-1}S)$, so by induction hypothesis we get the following $d_1(y) \in \mathrm{Pred}(F_1)(\neg S)$ and $d_2(y) \in \mathrm{Pred}(F_2)(\neg S)$, and then $d(y) \in \mathrm{Pred}(F)(\neg S)$.
4. $F = F_1 + F_2$. Without loss of generallity, let us suppose that $c(x) = \kappa_1(c_1(x))$, that is, $c(x) = (c_1(x), 1)$, so we have $c_1(x) \in \mathrm{Pred}(F_1)(\neg R^{-1}S)$. Let us consider the constant coalgebras:

$$\begin{array}{rl} c_X : X \to F_1X & \qquad d_Y : Y \to F_1Y \\ z \;\mapsto c_1(x) & \qquad z \;\mapsto d_1(y) \end{array}$$

   Trivially, R is a bisimulation between c_X and d_Y; applying the induction hypothesis we get $d_1(y) \in \mathrm{Pred}(F_1)(\neg S)$ and hence $d(y) \in \mathrm{Pred}(F)(\neg S)$. Reasoning in an analogous way we get that if $c(x) = \kappa_2(c_1(x))$, also $d(y) \in \mathrm{Pred}(F)(\neg S)$.
5. $F = G^A$.

$$\mathrm{Pred}(F)(\neg R^{-1}S) = \{f \mid \forall a \in A.\, \mathrm{Pred}(G)(\neg R^{-1}S)(f(a))\}\,.$$

   Now, for each $a \in A$ and any F-coalgebra $c : X \longrightarrow FX$, we can define a new G-coalgebra: $c^a : X \longrightarrow GX$ where, for each $x \in X$ we have $c^a(x) = c(x)(a)$; and analogously we define $d^a(y) = c(y)(a)$ for each $y \in Y$. This way we have xRy and $c^a(x) = c(x)(a) \in \mathrm{Pred}(G)(\neg R^{-1}S))$. Using the induction hypothesis we get $d^a(y) \in \mathrm{Pred}(G)(\neg S)$. This argument is valid for each $a \in A$, so we get $d(y) \in \mathrm{Pred}(F)(\neg S)$.

6. $F = \mathcal{P}(G)$. In this case

$$\mathrm{Pred}(F)(\neg R^{-1}S) = \{U \mid \forall u \in U.\, \mathrm{Pred}(G)(\neg R^{-1}S)(u)\}\,.$$

We have $c(x) \in \mathrm{Pred}(F)(\neg R^{-1}S)$, so for all $a \in c(x)$ is true that $a \in \mathrm{Pred}(G)(\neg R^{-1}S)$ and we want to prove that $d(y) \in \mathrm{Pred}(F)(\neg S)$ or, equivalently, that for all $b \in d(y)$ we have $b \in \mathrm{Pred}(G)(\neg S)$. Let us take one $b \in d(y)$ and we define the constant coalgebra:

$$\begin{aligned} d^b_Y : Y &\to GY \\ z &\mapsto b \end{aligned}$$

Now, from our hypothesis xRy, and R is a bisimulation, so we also have $c(x)\mathrm{Rel}(F)(R)d(y)$, and by definition of relation lifting it follows that there exists some $a \in c(x)$ such that $a\mathrm{Rel}(G)(R)b$.
Now, we consider another constant coalgebra:

$$\begin{aligned} c^a_X : X &\to GX \\ z &\mapsto a \end{aligned}$$

Trivially R is a bisimulation between the coalgebras c^a_X and d^b_Y because $a\mathrm{Rel}(G)(R)b$; also $c^a_X(x) = a \in \mathrm{Pred}(G)(\neg R^{-1}S)$, so by induction hypothesis it follows that $d^b_Y(y) = b \in \mathrm{Pred}(G)(\neg S)$. Since this argument is valid for all $b \in d(y)$ we have proved that $d(y) \in \mathrm{Pred}(F)(\neg S)$.

7. $F = G^*$. As we have previously done, let us suppose the following $c(x) = \langle c_1(x), \ldots, c_n(x)\rangle$ and $d(y) = \langle d_1(y), \ldots, d_n(y)\rangle$. Then, for each $i \leq n$ we have $c_i(x) \in \mathrm{Pred}(G)(\neg R^{-1}S)$. Now, we define these two families of constant coalgebras:

$$\begin{aligned} c^x_i : X &\to GX \\ z &\mapsto c_i(x) \end{aligned} \qquad\qquad \begin{aligned} d^y_i : Y &\to GY \\ z &\mapsto d_i(y) \end{aligned}$$

Trivially, R is a bisimulation between c^x_i and d^y_i for each $i \leq n$. Therefore, using the induction hypothesis we get $d_i(y) \in \mathrm{Pred}(G)(\neg S)$ for each $i \leq n$, that is, $d(y) \in \mathrm{Pred}(F)(\neg S)$.

□

At last we can prove Lemma 4:

*Proof (***Lemma 4***).* Once again we use the following notation:

$$f_1 \quad \text{denotes} \quad f^U_{([\![\varphi_1]\!],[\![\varphi_2]\!])}(S)\,,$$

$$f_2 \quad \text{denotes} \quad f^U_{(R^{-1}[\![\varphi_1]\!],R^{-1}[\![\varphi_2]\!])}(S)\,.$$

Let us suppose $y \in [\![\varphi_1]\!]\mathcal{U}[\![\varphi_2]\!]$, that is, $y \in \mu S.f_1(S)$, and we have to show that $x \in \mu S.f_2(S)$. By monotonicity and continuity of $f^U_{(P,Q)}$, their least fixed points are given by

$$\mu S.f_1(S) = \bigcup_{i=0}^{\infty} f_1^i(\emptyset)\,,$$

$$\mu S.f_2(S) = \bigcup_{i=0}^{\infty} f_2^i(\emptyset)\,.$$

So, since $y \in \bigcup_i^\infty f_1^i(\emptyset)$ we have $y \in f_1^i(\emptyset)$ for some i. Henceforth, $x \in R^{-1}f_1^i(\emptyset)$. Also, by Lemma 7 we have

$$x \in R^{-1}f_1^i(\emptyset) \subseteq f_2(R^{-1}f_1^{i-1}(\emptyset))\,,$$

where by monotonicity of f_2,

$$f_2(R^{-1}f_1^{i-1}(\emptyset)) \subseteq f_2(f_2(R^{-1}f_1^{i-2}(\emptyset)))\,.$$

If we iterate this process we finally get

$$x \in f_2^i(R^{-1}\emptyset) = f_2^i(\emptyset)\,.$$

And, that way, we get $x \in \bigcup_i^\infty f_2^i(\emptyset) = \mu S.f_2(S)$. □

With this lemma, the proposition involving reflection and the *until* operator is immediate.

Proposition 7. *Let φ_1 and φ_2 be temporal formulas such that they are reflected by bisimulations, $\psi = \varphi_1\ \mathcal{U}\ \varphi_2$ a temporal formula on Y and R a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then, the property ψ is reflected by R.*

Proof. Let aRb and let us suppose $b \models \varphi_1\ \mathcal{U}\ \varphi_2$, that is, $b \in [\![\varphi_1]\!]\ \mathcal{U}\ [\![\varphi_2]\!]$. By Lemma 4 we also get $a \in R^{-1}[\![\varphi_1]\!]\ \mathcal{U}\ R^{-1}[\![\varphi_2]\!]$, and from both monotony of the operator *until* and Lemma 5 we get $a \in [\![\varphi_1^{-1}]\!]\ \mathcal{U}\ [\![\varphi_2^{-1}]\!]$. □

Preservation of properties is a consequence of the reflection of properties together with the fact that if R is a bisimulation then R^{-1} is also a bisimulation. We have thus proved the following theorem.

Theorem 1. *Let ψ and φ be formulas over sets Y and X, respectively, which do not use the* negation *operator and let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Then ψ is reflected by R and φ is preserved by R.*

4 Reflection and preservation in simulations

In [3,15] it is proved not only that bisimulations reflect and preserve properties but also that simulations reflect them: it turns out that this result does not generalize straightforwardly to the coalgebraic setting.

The main problem that we have found concerning this is that the coalgebraic definition of simulation uses an arbitrary functorial order $\sqsubseteq$, and in general reflection of properties will not hold for all orders.

Let us show a counterexample that will convince us that simulations may not reflect properties without restricting the orders. Let us take $F = \mathcal{P}(id)$, $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$ and the coalgebras c and d defined as $c(x_1) = \{x_1, x_2\}$, $c(x_2) = \{x_2\}$, $d(y_1) = y_2$ and $d(y_2) = y_2$. We define $u \sqsubseteq v$ whenever $v \subseteq u$ and consider the formula $\varphi = \bigcirc P$, where $P = \{y_2\}$, and the simulation $R = \{(x_1, y_2)\}$. It is immediate to check that R is a simulation and $y_2 \in [\![\varphi]\!]$, but $x_1 \notin [\![\varphi^{-1}]\!]$.

- $y_2 \in [\![\varphi]\!]$. Indeed, since $d(y_2) = y_2$ then $y_2 \in [\![\varphi]\!] = \bigcirc P$ is equivalent to $y_2 \in P = \{y_2\}$, which is trivially true.
- $x_1 \notin [\![\varphi^{-1}]\!]$. By definition, $\varphi^{-1} = \bigcirc R^{-1} P = \bigcirc\{x_1\}$. Since $c(x_1) = \{x_1, x_2\}$, it is enough to see that $x_2 \notin \{x_1\}$, which is also true.

As a consequence, we will need to restrict the functorial orders that are involved in the definition of simulation. In a first approach we will impose an extra requirement that the order must fulfill, and later we will not only restrict the orders but also the functors that are involved.

4.1 Restricting the orders

The idea is that we are going to require an extra property for each pair of elements which are related by the order. In particular, we are particularly interested in the following property (which is defined in [7]):

Definition 5. *Given a functor F : **Sets** $\longrightarrow$ **Sets**, we say that an order $\sqsubseteq$ associated to it is "down-closed" whenever $a \sqsubseteq b$, with $a, b \in FX$, implies that*

$$b \in \mathrm{Pred}(F)(P) \implies a \in \mathrm{Pred}(F)(P), \quad \textit{for all predicates } P \subseteq X\,.$$

We can show some examples of down-closed orders:

Example 2. 1. Kripke structures are defined by the functor $F = \mathcal{P}(AP) \times \mathcal{P}(id)$, so a down-closed order must fulfill that if $(u, v) \sqsubseteq (u', v')$, then $(u', v') \in \mathrm{Pred}(F)(P)$ implies $(u, v) \in \mathrm{Pred}(F)(P)$; that is, by definition of $\mathrm{Pred}(\mathcal{P}(AP) \times \mathcal{P}(id))$, $u, u' \subseteq \mathcal{P}(AP)$ and, if $v' \in \mathrm{Pred}(\mathcal{P}(id))(P) = \{U \mid \forall u \in U.\, u \in P\}$ then $v \in \mathrm{Pred}(\mathcal{P}(id))(P)$. In other words, for all $b \in v$ and $b' \in v'$, if $b' \in P$ then $b \in P$. Therefore, what is needed in this case is that the set of successors v of the smaller pair is contained in the set of successors v' of the bigger pair, that is,

$$\text{if} \quad (u, v) \sqsubseteq (u', v') \quad \text{then} \quad v \subseteq v'\,.$$

2. Labelled transition systems are defined by the functor $F = \mathcal{P}(id)^A$, so the order must fulfill the following:

$$\text{if} \quad u \sqsubseteq v \quad \text{then} \quad \forall a \in A.\, u(a) \subseteq u'(a)\,.$$

These examples show that there are not many down-closed orders, but it does not seem clear how to further extend this class in such a way that we could still prove the reflection of properties by simulations. Unfortunately, even under this restriction we can only prove reflection (or preservation) of formulas that only use the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$.

To convince us of this fact, we present a counterexample with operator $\Diamond$. Let $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$ and the functor $F = \mathcal{P}(id)$. We consider the following down-closed order: $u \sqsubseteq v$ if $u \subseteq v$. We also define the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ as $c(x_1) = \{x_1\}$, $c(x_2) = \{x_2\}$, $d(y_1) = \{y_1, y_2\}$ and $d(y_2) = \{y_2\}$. Obviously $R = \{(x_1, y_1)\}$ is a simulation since $c(x_1) = \{x_1\} \sqsubseteq \{x_1\}$ and $\{y_1\} \sqsubseteq \{y_1, y_2\} = d(y_1)$ and, also, $\{x_1\}\mathrm{Rel}(F)(R)\{y_1\}$. We have $y_1 \in \Diamond\{y_2\}$, since we can reach y_2 from y_1, but $x_1 \notin \Diamond R^{-1}\{y_2\} = \Diamond\emptyset$. Indeed, $x_1 \notin \Diamond\emptyset$ is equivalent to $x_1 \in \Box\neg\emptyset$ and this is true since $\{x_1\}$ is an invariant such that $x_1 \in \{x_1\}$, with $\{x_1\} \subseteq \neg\emptyset$.

In order to prove reflection of properties that only use the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, we will need a previous elementary result involving binary relations.

Proposition 8. *Let $R \subseteq X \times Y$ be a binary relation and $P \subseteq Y$ a predicate. Let us suppose that $u\mathrm{Rel}(F)(R)v$; then if $v \in \mathrm{Pred}(F)(P)$ it is also true that $u \in \mathrm{Pred}(F)(R^{-1}P)$.*

Proof. Once again the proof will proceed by structural induction on the functor F.

1. If F is constant, then the result follows trivially.
2. Let us suppose that $F = id$, then we have uRv and also $v \in P$ and therefore, by definition of $R^{-1}P$, we get $u \in R^{-1}P$.
3. Let us now suppose that $F = F_1 \times F_2$ and let $u = (u_1, u_2)$ and $v = (v_1, v_2)$. By definition of the relation lifting we have both $u_1\mathrm{Rel}(F_1)(R)v_1$ and $u_2\mathrm{Rel}(F_2)(R)v_2$; whereas by definition of predicate lifting, since $v \in \mathrm{Pred}(F)(P)$ we have $v_1 \in \mathrm{Pred}(F_1)(P)$ and $v_2 \in \mathrm{Pred}(F_2)(P)$. So, applying the induction hypothesis we get $u_1 \in \mathrm{Pred}(F_1)(R^{-1}P)$ and $u_2 \in \mathrm{Pred}(F_2)(R^{-1}P)$, henceforth $u \in \mathrm{Pred}(F)(R^{-1}P)$.
4. If $F = F_1 + F_2$, without loss of generality let us suppose $v = \kappa_1(v_0)$ and $u = \kappa_1(u_0)$. Then, by the definition of predicate lifting we have $v_0 \in \mathrm{Pred}(F_1)(P)$. Also, by the definition of the relation lifting $u_0\mathrm{Rel}(F_1)(R)v_0$, so by induction hypothesis we obtain $u_0 \in \mathrm{Pred}(F_1)(R^{-1}P)$, that is, $u \in \mathrm{Pred}(F)(R^{-1}P)$.
5. Let us suppose $F = G^A$. If $v \in \mathrm{Pred}(F)(P)$ then for all $a \in A$ we will have $v(a) \in \mathrm{Pred}(G)(P)$. But, on the other hand, since $u\mathrm{Rel}(F)(R)v$ then for all $a \in A$ it is also true that $u(a)\mathrm{Rel}(G)(R)\ v(a)$. Let us consider any $a_0 \in A$; then $v(a_0) \in \mathrm{Pred}(G)(P)$ and $u(a_0)\ \mathrm{Rel}(G)(R)v(a_0)$, so by induction hypothesis we get $u(a_0) \in \mathrm{Pred}(G)(R^{-1}P)$. This is valid for any $a_0 \in A$, so it proves that $u \in \mathrm{Pred}(F)(R^{-1}P)$.
6. Let us suppose $F = \mathcal{P}(G)$. In this case, since $v \in \mathrm{Pred}(F)(P)$ we have that for each $b \in v$, then $b \in \mathrm{Pred}(G)(P)$. Our goal is to show that $u \in \mathrm{Pred}(F)(R^{-1}P)$, that is, for all $a \in u$ it must be $a \in \mathrm{Pred}(G)(R^{-1}P)$. Let us take any $a \in u$, since $u\mathrm{Rel}(F)(R)v$ there exists $b \in v$ such that $a\mathrm{Rel}(G)(R)b$.

By induction hypothesis we get $a \in \mathrm{Pred}(G)(R^{-1}P)$; since this is a valid argument for all $a \in u$, it follows that $u \in \mathrm{Pred}(F)(R^{-1}P)$.

7. Let us suppose $F = G^*$, $v = \langle v_1, \ldots, v_n \rangle$ and $u = \langle u_1, \ldots, u_n \rangle$. Then, since $v \in \mathrm{Pred}(F)(P)$ for each $i \leq n$ we have that $v_i \in \mathrm{Pred}(G)(P)$. By the definition of the relation lifting we have that for each $i \leq n$ then $u_i \mathrm{Rel}(G)(R) v_i$, hence by induction hypothesis, for all $i \leq n$ it follows that $u_i \in \mathrm{Pred}(G)(R^{-1}P)$ and therefore $u \in \mathrm{Pred}(F)(R^{-1}P)$.

□

We will also need a subtle adaptation of Lemmas 3 and 5 from the framework of bisimulations to the framework of simulations. In particular, we can adapt Lemma 3 to prove that if Q is an invariant and R a simulation, $R^{-1}Q$ is still an invariant, whereas Lemma 5 will also be true in the framework of simulations for formulas that only use the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$.

Lemma 8. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, with a down-closed order, and let $Q \subseteq Y$ be an invariant. Then $R^{-1}Q$ is also an invariant.*

Proof. We are going to show that for all $x \in R^{-1}Q$, we also have $c(x) \in \mathrm{Pred}(F)(R^{-1}Q)$. Let us take an arbitrary $x \in R^{-1}Q$; then, by definition there exists $y \in Q$ such that xRy and, since Q is an invariant, $d(y) \in \mathrm{Pred}(F)(Q)$. On the other hand, since R is a simulation, $c(x) \sqsubseteq u\mathrm{Rel}(F)(R)v \sqsubseteq d(y)$. Henceforth, since we are working with a down-closed order and $d(y) \in \mathrm{Pred}(F)(Q)$, then $v \in \mathrm{Pred}(F)(Q)$. Also, by Proposition 8 we have $u \in \mathrm{Pred}(F)(R^{-1}Q)$ and, using again that the order is down-closed, it follows that $c(x) \in \mathrm{Pred}(F)(R^{-1}Q)$.

Lemma 9. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, with a down-closed order. If φ is a temporal formula constructed only with operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, then*

$$R^{-1}[\![\varphi]\!]_Y \subseteq [\![\varphi^{-1}]\!]_X \,.$$

Proof. This time the proof will proceed by structural induction on the formula φ.

1. $\varphi = P$, where $P \subseteq Y$ is a predicate. Clearly, by definition we have that $R^{-1}[\![\varphi]\!] = R^{-1}P = [\![\varphi^{-1}]\!]$.
2. $\varphi = \varphi_1 \vee \varphi_2$, or $\varphi = \varphi_1 \wedge \varphi_2$. In both cases we trivially have the result.
3. $\varphi = \bigcirc\varphi_1$. Let us suppose that $x \in R^{-1}[\![\bigcirc\varphi_1]\!]$; henceforth there exists some y such that xRy with $y \in [\![\bigcirc\varphi_1]\!] = \bigcirc[\![\varphi_1]\!]$. Equivalently we have xRy with $d(y) \in \mathrm{Pred}(F)([\![\varphi_1]\!])$ and since R is a simulation we also get that $c(x) \sqsubseteq u\mathrm{Rel}(F)(R)v \sqsubseteq d(y)$. Using the fact that $\sqsubseteq$ is down-closed and Proposition 8 we obtain $c(x) \in \mathrm{Pred}(F)(R^{-1}[\![\varphi_1]\!])$. Now, by induction hypothesis we know that $R^{-1}[\![\varphi_1]\!] \sqsubseteq [\![\varphi_1^{-1}]\!]$; this, together with the monotonicity of the operator $\mathrm{Pred}(F)$ leads us to $c(x) \in \mathrm{Pred}(F)([\![\varphi_1^{-1}]\!])$, that is, $x \in [\![\varphi^{-1}]\!]$.

4. $\varphi = \Box\varphi_1$. By definition, $\Box[\![\varphi_1]\!]$ is the greatest invariant contained in $[\![\varphi_1]\!]$ henceforth, $R^{-1}[\![\varphi_1]\!]$ is also an invariant. Trivially, since $\Box[\![\varphi_1]\!] \subseteq [\![\varphi_1]\!]$ we have $R^{-1}\Box[\![\varphi_1]\!] \subseteq R^{-1}[\![\varphi_1]\!]$. By induction hypothesis, $R^{-1}[\![\varphi_1]\!] \subseteq [\![\varphi_1^{-1}]\!]$, so $R^{-1}\Box[\![\varphi_1]\!]$ is an invariant contained in $[\![\varphi_1^{-1}]\!]$ and hence it must be contained in the greatest invariant contained in $[\![\varphi_1^{-1}]\!]$, that is, it must be contained in $\Box[\![\varphi_1^{-1}]\!]$, as we wanted to prove.

□

Now we can state and prove the corresponding theorem:

Theorem 2. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ with a down-closed order. If φ is a temporal formula constructed only with operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, then the property φ is reflected by the simulation.*

Proof. Let us suppose xRy; it will be enough to prove that $y \in [\![\varphi]\!]$ implies $x \in R^{-1}[\![\varphi]\!]$. As in the previous proofs, we use structural induction on the formula φ.

1. If $\varphi = P \subseteq Y$ is an arbitrary predicate then we must prove that $y \in P$ implies $x \in R^{-1}P$. Since xRy, the result follows.
2. Let us suppose $\varphi = \varphi_1 \vee \varphi_2$ and let $y \in [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$. Without loss of generality we take $y \in [\![\varphi_1]\!]$. Then, since xRy we have $x \in R^{-1}[\![\varphi_1]\!] \subseteq R^{-1}[\![\varphi_1]\!] \cup R^{-1}[\![\varphi_2]\!]$, as required.
3. The case $\varphi = \varphi_1 \wedge \varphi_2$ is similar to the previous case.
4. Let $\varphi = \bigcirc\varphi_1$. If $y \in \bigcirc[\![\varphi_1]\!]$, by definition we have $d(y) \in \mathrm{Pred}(F)([\![\varphi_1]\!])$. Since xRy and R is a simulation, $c(x) \sqsubseteq u\mathrm{Rel}(F)(R)v \sqsubseteq d(y)$ and using that $\sqsubseteq$ is down-closed and Proposition 8 it follows that $c(x) \in \mathrm{Pred}(F)(R^{-1}[\![\varphi_1]\!])$.
5. Let us suppose $\varphi = \Box\varphi_1$: by definition, there is an invariant Q for the coalgebra d such that $y \in Q$ and $Q \subseteq [\![\varphi_1]\!]$. We must prove that there exists an invariant S for c such that $x \in S$ with $S \subseteq R^{-1}[\![\varphi_1]\!]$. Let us take the invariant S defined as $S = R^{-1}Q$ which trivially contains x. Now, since $Q \subseteq [\![\varphi_1]\!]$, we have $S = R^{-1}Q \subseteq R^{-1}[\![\varphi_1]\!]$, as required.

□

Instead of considering down-closed orders, we could have imposed the converse implication, that is, those orders that satisfy that if $a \in \mathrm{Pred}(F)(P)$ then $b \in \mathrm{Pred}(F)(P)$.

Definition 6. *Given a functor F : **Sets** $\longrightarrow$ **Sets** we say that an order $\sqsubseteq$ is* up-closed *if whenever $a \sqsubseteq b$ then*

$$a \in \mathrm{Pred}(F)(P) \implies b \in \mathrm{Pred}(F)(P), \quad \textit{for all predicates } P\,.$$

Obviously up-closed is symmetrical to down-closed, that is, it is equivalent to taking $\sqsubseteq^{op}$ instead of $\sqsubseteq$ in Definition 5. So, for example, in the case of Kripke structures an up-closed order would satisfy

$$(u, v) \sqsubseteq (u', v') \quad \text{if} \quad v' \subseteq v\,.$$

The interesting thing about up-closed orders is that they allow us to prove *preservation* of properties; again, this result will hold only for formulas constructed with the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$. We need the following auxiliary result whose proof is analogous to the case of down-closed orders.

It is well-known that if R is a simulation for the order $\sqsubseteq$, then R^{-1} is a simulation for the oposite order $\sqsubseteq^{op}$. Using this property we get the following:

Theorem 3. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ carrying an up-closed order. If φ is a temporal formula constructed only with the operators $\vee$, $\wedge$, $\bigcirc$ and $\Box$, then R preserves the property φ.*

Proof. Let us suppose xRy and $x \in [\![\varphi]\!]$. Let us consider $S = R^{-1}$. We know that S is a simulation between $d : Y \longrightarrow FY$ and $c : X \longrightarrow FX$ with the down-closed order $\sqsubseteq^{op}$ and since $(x, y) \in R$ then, $(y, x) \in S$. Hence, we can apply Theorem 2 and since $x \in [\![\varphi]\!]$ then, $y \in [\![\varphi^{-1}]\!]$. But when considering $S^{-1} = R$ the latter is the same as $y \in [\![\varphi^*]\!]$ (remember Remark 1). Hence we have proved that if $x \in [\![\varphi]\!]$ then, $y \in [\![\varphi^*]\!]$, that is, the preservation of the property. $\Box$

4.2 Restricting the class of functors

As we have just seen, it is not enough to restrict ourselves to down-closed (or up-closed) orders to get a valid result for all properties. What we want is a necessary and sufficient condition over functorial orders that implies reflection (or preservation) of properties by simulations. So far we have not found such a condition, but we have a sufficient one for simulations to reflect properties (and, in fact, also so that they preserve properties).

Recalling the structure of lemmas and propositions used to prove reflection and preservation of properties by bisimulations, we notice that the key ingredients were Lemmas 2 and 1. With these lemmas we were able to prove directly preservation of invariants (Lemma 3) and the relation between R^{-1} (respectively R) of a formula and the inverse of a formula (respectively direct image of a formula). Also, Lemmas 2 and 1 were essential to prove directly reflection and preservation of formulas built with the *nexttime* operator and the rest of temporal operators.

In the previous section the problem we faced was that either Lemma 2 (for down-closed orders) or Lemma 1 (for up-closed orders) held, but not both simultaneously. As a consequence, the results for the operators *eventually* and *until* did not hold. So, if we were capable of finding a subclass of functors and orders such that they fulfill results analogous to Lemmas 2 and 1 then, translating those proofs, we would get reflection and preservation of arbitrary properties.

We are going to define a subclass of functors and orders in the way that Hughes and Jacobs did in [7] for the subclass **Poly**.

Definition 7. *The class* **Order** *is the least class of functors closed under the following:*

1. *For every preorder $(A, \leq)$, the constant functor $X \mapsto A$ with the order given by $\sqsubseteq_X = \leq_A$ is in* **Order**.

2. *The identity functor with equality order is in* **Order**.
3. *Given two polynomial functors F_1 and F_2 with orders $\sqsubseteq^1$ and $\sqsubseteq^2$, the product functor $F_1 \times F_2$ with order $\sqsubseteq_X$ given by*

$$(u, v) \sqsubseteq_X (u', v') \quad \text{if} \quad u \sqsubseteq^1 u' \quad \text{and} \quad v \sqsubseteq^2 v' \,,$$

   is in **Order**.
4. *Given the polynomial functor F with order $\sqsubseteq^F$ and the set A, the functor F^A with order $\sqsubseteq_X$ given by*

$$u \sqsubseteq_X v \text{ if } u(a) \sqsubseteq^F v(a) \text{ for all } a \in A,$$

   is in **Order**.
5. *Given two polynomial functors F_1 and F_2 with orders $\sqsubseteq^1$ and $\sqsubseteq^2$, the co-product functor $F_1 + F_2$ with order $\sqsubseteq_X$ given by*

$$\begin{array}{r} u \sqsubseteq_X v \text{ if } u = \kappa_1(u_0) \text{ and } v = \kappa_1(v_0) \text{ with } u_0 \sqsubseteq^1 v_0 \\ \text{or } u = \kappa_2(u_0) \text{ and } v = \kappa_2(v_0) \text{ with } u_0 \sqsubseteq^2 v_0 \,, \end{array}$$

   is in **Order**.
6. *Given the polynomial functor F with order $\sqsubseteq^F$, the powerset functor $\mathcal{P}(F)$ with order $\sqsubseteq_X$ given by*

$$\begin{array}{rlll} u \sqsubseteq_X v & \text{if} & \forall a \in u\ \exists b \in v & \text{such that} \quad a \sqsubseteq^F b \\ & \text{and also} & \forall b \in v\ \exists a \in u & \text{such that} \quad a \sqsubseteq^F b \,, \end{array}$$

   is in **Order**.

For example the usual order for Kripke structures is not in the class **Order**. Besides, in the definition of **Poly** in [7] the authors did not consider the powerset functor but we do, although we are not using the *usual* order for this functor.

At first, to obtain that simulations not only reflect but also preserve properties may seem a little surprising. If we think about the elements in the subclass **Order** we notice that we have restricted the orders to equality-like orders, that is, almost all possible orders in **Order** are the equality. However, since the class **Order** is very similar to the class **Poly**, it has the same good properties shown in [7] (like the stability of the orders and functors). Let us see some orders and functors that belong to **Order**:

Example 3. 1. If we consider the functor $\mathcal{P}(id)$, then the order $\sqsubseteq$ defined in Definition 7 says that $u \sqsubseteq v$ if and only if for each $a \in u$ there exists $b \in v$ such that $a = b$, and if for each $b \in v$ there exists $a \in u$ such that $a = b$. This means that $\sqsubseteq$ is the identity relation. As an immediate consequence for transition systems the only possible **Order** simulations are bisimulation.
2. If we consider the functor $A \times id$ where A has a preorder $\leq_A$ different from the identity, the order $\sqsubseteq$ from Definition 7 is the following: $(u, v) \sqsubseteq (u', v')$ iff $v = v'$ and $u \leq_A u'$. So, if $\leq_A$ is not the identity, neither is $\sqsubseteq$. For example, let us take $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2\}$, $AP = \{p_1, p_2, p_3\}$ and consider

the functor $F = \mathcal{P}(id) \times \mathcal{P}(AP)$ and the coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ defined by $c(x_1) = (\{x_2, x_3\}, \{p_1\})$, $c(x_2) = (\{x_3\}, \{p_2\})$, $c(x_3) = (\{x_2\}, \{p_3\})$, $d(y_1) = (\{y_2\}, \{p_2\})$ and $d(y_2) = (\{y_2\}, \{p_1\})$. Obviously there is no bisimulation between x_1 and y_1 since this atomic propositions are not the same, but taking the order $\sqsubseteq$ defined as $(u, v) \sqsubseteq (u', v')$ iff $u = u'$ (that is, taking as the preorder $\leq_{AP}$ the total relation) we have that there exists a simulation R in **Order** between x_1 and y_1.

Lemma 10. *Let $R \subseteq X \times Y$ be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, such that the functor F is in the class* **Order**. *Let us also suppose that $P \subseteq Y$ and xRy; then, if $d(y) \in \mathrm{Pred}(F)(P)$ we have $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.*

Proof. Again, the proof will be done by structural induction on the functor F.

1. $F = cnst$. In this case $\mathrm{Pred}(F)(P) = \top$ and also $\mathrm{Pred}(F)(R^{-1}P) = \top$, and $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$ trivially.
2. $F = id$. In this case we have that the order $\sqsubseteq$ coincides with the equality and $\mathrm{Pred}(F)(P) = P$ for each P. Henceforth, we must check that $c(x) \in R^{-1}P$. Since R is a simulation we have $c(x) \sqsubseteq uRv \sqsubseteq d(y)$ and this is equivalent to $c(x)Rd(y)$, because the order is the equality.
3. Let F_1 and F_2 have orders $\sqsubseteq^1$ and $\sqsubseteq^2$ and consider $F = F_1 \times F_2$ with the order defined in Def. 7. We have

$$\mathrm{Pred}(F)(P) = \{(u, v) \mid \mathrm{Pred}(F_1)(P)(u) \wedge \mathrm{Pred}(F_2)(P)(v)\}\,.$$

   Let us suppose that $d(y) = (d_1(y), d_2(y))$ and similarly $c(x) = (c_1(x), c_2(x))$. Then, if $d(y) \in \mathrm{Pred}(F)(P)$ we have $d_1(y) \in \mathrm{Pred}(F_1)(P)$ and $d_2(y) \in \mathrm{Pred}(F_2)(P)$. Now, as R is a simulation between c and d, from xRy it follows the existence of $u = (u_1, u_2)$ and $v = (v_1, v_2)$ such that $c(x) \sqsubseteq (u_1, u_2)\mathrm{Rel}(F)(R)(v_1, v_2) \sqsubseteq d(y)$. By definition of $\sqsubseteq$, in particular we have both $c_1(x) \sqsubseteq^1 u_1\mathrm{Rel}(F_1)(R)v_1 \sqsubseteq^1 d_1(y)$ and $c_2(x) \sqsubseteq^2 u_2\mathrm{Rel}(F_2)(R)v_2 \sqsubseteq^2 d_2(y)$. That is, R is also a simulation between c_1 and d_1, and c_2 and d_2. Thus, we can use our induction hypothesis and since $d_1(y) \in \mathrm{Pred}(F_1)(P)$, we get $c_1(x) \in \mathrm{Pred}(F_1)(R^{-1}P)$ and, analogously, $c_2(x) \in \mathrm{Pred}(F_2)(R^{-1}P)$, so $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$, as we wanted to prove.
4. Let F_1 and F_2 have orders $\sqsubseteq^1$ and $\sqsubseteq^2$ and consider $F = F_1 + F_2$ with the order given by Def. 7. In this case, we have

$$\mathrm{Pred}(F)(P) = \{\kappa_1(u) \mid \mathrm{Pred}(F_1)(P)(u)\} \cup \{\kappa_2(v) \mid \mathrm{Pred}(F_2)(P)(v)\}.$$

   Without loss of generality we suppose $d(y) = \kappa_1(d_1(y)) = (d_1(y), 1)$; we must have $d_1(y) \in \mathrm{Pred}(F_1)(P)$. Let us consider the following constant coalgebras:

$$\begin{array}{rl} c_X : X \to F_1X & \qquad d_Y : Y \to F_1Y \\ z \mapsto c_1(x) & \qquad z \mapsto d_1(y) \end{array}$$

   Since R is a simulation and the order is the disjoint sum, R is also a simulation between c_X and d_Y. Applying the induction hypothesis, we have that $c_1(x) \in \mathrm{Pred}(F_1)(R^{-1}P)$ and hence $c(x) \in \mathrm{Pred}(F)(R^{-1}P)$.

5. Let F be a functor with orden $\sqsubseteq^F$ and consider the functor F^A with the order given by Def. 7. In this case,

$$\mathrm{Pred}(F^A)(P) = \{f \mid \forall a \in A.\, \mathrm{Pred}(F)(P)(f(a))\}\,,$$

$$\mathrm{Rel}(F^A)(R) = \{(f,g) \mid \forall a \in A.\, \mathrm{Rel}(F)(R)(f(a),g(a))\}\,.$$

Hence, there exists u and v such that $c(x) \sqsubseteq u\mathrm{Rel}(F^A)(R)v \sqsubseteq d(y)$. Now, for each $a \in A$ and F^A-coalgebra $d : Y \longrightarrow F^A(Y)$ we can define a coalgebra on F: $d^a : Y \longrightarrow F(Y)$ where, for each $y \in Y$, $d^a(y) = d(y)(a)$; analogously we define $c^a(x) = c(x)(a)$ for each $x \in X$. In this way we have that xRy and $d^a(y) = d(y)(a) \in \mathrm{Pred}(F)(P)$.
Now, using the definition of the order $\sqsubseteq$ we have the following

$$c^a(x) \sqsubseteq^F u(a)\mathrm{Rel}(F)(R)v(a) \sqsubseteq^F d^a(y)\,,$$

that is, R is also a simulation between c^a and d^a. Applying the induction hypothesis we get $c^a(x) \in \mathrm{Pred}(F)(R^{-1}P)$. Since this argument is valid for all $a \in A$, we finally get $c(x) \in \mathrm{Pred}(F^A)(R^{-1}P)$.

6. Let F be a functor with order $\sqsubseteq^F$ and let us consider the functor $\mathcal{P}(F)$ with the order given by Def. 7. In this case

$$\mathrm{Pred}(\mathcal{P}(F))(P) = \{U \mid \forall u \in U.\, \mathrm{Pred}(F)(P)(u)\}\,.$$

We have $d(y) \in \mathrm{Pred}(\mathcal{P}(F))(P)$ so for each $b \in d(y)$ we have that $b \in \mathrm{Pred}(F)(P)$, and we must prove that $c(x) \in \mathrm{Pred}(\mathcal{P}(F))(R^{-1}P)$, or equivalently, that for all $a \in c(x)$ also $a \in \mathrm{Pred}(F)(R^{-1}P)$. Let us take an arbitrary $a \in c(x)$, and we define the following constant coalgebra:

$$\begin{aligned} c^a_X : X &\to FX \\ z &\mapsto a \end{aligned}$$

Now, since xRy and R is a simulation, $c(x) \sqsubseteq u\mathrm{Rel}(\mathcal{P}(F))(R)v \sqsubseteq d(y)$. By definition of $\sqsubseteq$, from $c(x) \sqsubseteq u$ it follows that for each $a \in c(x)$ there exists some $a_1 \in u$ such that $a \sqsubseteq^F a_1$. Also, by the definition of the relation lifting we have that for each element $a_1 \in u$ there exists $b_1 \in v$ such that $a_1\mathrm{Rel}(F)(R)b_1$. Again by the definition of the order, for each $b_1 \in V$ there exists a $b \in d(y)$ such that $b_1 \sqsubseteq^F b$.
Now we consider:

$$\begin{aligned} d^b_Y : Y &\to FY \\ z &\mapsto b \end{aligned}$$

Trivially, R is a simulation between the coalgebras c^a_X and d^b_Y, because $a \sqsubseteq^F a_1\mathrm{Rel}(G)(R)b_1 \sqsubseteq^F b$; also $d^b_Y = b \in \mathrm{Pred}(F)(P)$, so by induction hypothesis it follows that $c^a_X = a \in \mathrm{Pred}(F)(R^{-1}P)$. Since this argument is valid for each $a \in c(x)$ we get $c(x) \in \mathrm{Pred}(\mathcal{P}(F))(R^{-1}P)$.

□

In a similar way we have the corresponding lemma involving direct predicates.

Lemma 11. *Let $R \subseteq X \times Y$ be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, such that the functor F is in* **Order**. *Let us suppose also that $P \subseteq X$ and xRy. Then, if $c(x) \in \mathrm{Pred}(F)(P)$, $d(y) \in \mathrm{Pred}(F)(RP)$.*

Now we can conclude that under these hypothesis simulations reflect and preserve properties, simultaneously! This fact is a straightforward result from Lemmas 10 and 11.

Theorem 4. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$, with F a polynomial functor in the class* **Order**. *Then, the simulation R reflects and preserves properties.*

5 Including atomic propositions

A consequence of the fact that the logic proposed by Jacobs does not introduce atomic propositions was the need of giving non-standard definitions of reflection and preservation of properties. Kurz, in his work [12] includes atomic propositions in a temporal logic for coalgebras by means of natural transformations.

Definition 8. *Given a set AP of atomic propositions, the formulas of the temporal logic associated to a coalgebra $c : X \longrightarrow FX$ are given by the BNF expression:*

$$\varphi = p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi \,\mathcal{U}\, \varphi$$

where $p \in AP$ is an atomic proposition.

Kurz also defines when a state x satisfies an atomic proposition p, that is, he defines the semantics of an atomic proposition.

Definition 9. *Let F : **Sets** $\longrightarrow$ **Sets** be a functor and AP a set of atomic propositions. Let $\nu : F \Rightarrow \mathcal{P}(AP)$ be a natural transformation and $c : X \longrightarrow FX$ a coalgebra. We say that x satisfies an atomic proposition $p \in AP$, and denote it $x \models p$, when*

$$p \in (\nu_X \circ c)(x)\,.$$

This way $[\![p]\!] = \{x \mid p \in (\nu_X \circ c)(x)\}$.

Notice that in fact this defines not only a semantics but a family of possible semantics that depends on the natural transformation. For example, we can define a natural transformation for the functor for Kripke structures in this way:

$$\begin{array}{rccc} \nu_X : & \mathcal{P}(AP) \times \mathcal{P}(X) & \longrightarrow & \mathcal{P}(AP) \\ & (P, Q) & \longmapsto & P \end{array}$$

With ν_X we have characterized the standard semantics of LTL for Kripke structures. Analogously, we could define the following interpretation: $\nu'_X(P, Q) = \mathcal{P}(AP) \setminus P$.

Introducing in our temporal logic the semantics of the atomic propositions, we can state the following theorem involving bisimulations:

Theorem 5. *Let R be a bisimulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$. Let φ be a temporal formula; then, the following is true for all $x \in X$ and $y \in Y$ such that xRy:*

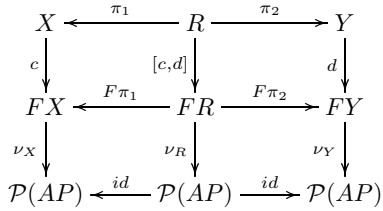
$$x \in [\![\varphi]\!]_X \quad \Longleftrightarrow \quad y \in [\![\varphi]\!]_Y \, .$$

Here we have captured in the same theorem the classical ideas of reflection and preservation of properties: we have some property in the lefthand side of a bisimulation if and only if we have the property in its righthand side. In this case the theorem is true also for the negation operator thanks to the atomic propositions. Intuitively, this is because now we have an "if and only if" theorem, whereas in Theorem 1 we needed to reason separately for each implication using monotonicity, and negation lacks it. Also notice that even though we could think that in Theorem 1 our predicates played the role of atomic propositions, there are some essential differences: first, predicates are not independent of each other, unlike atomic propositions, and secondly, while atomic propositions stay the same predicates vary with each set of states.

Now we prove the theorem:

Proof. Once again the proof will proceed by structural induction on the formula φ.

1. Let $\varphi = p$ where p is an arbitrary atomic proposition. This way we have the following diagram, for ν an arbitrary natural trasformation:



   This diagram is commutative. Indeed, since R is a bisimulation the upper side commutes, while the lower side commutes because ν is a natural transformation.
   So, $x \in [\![\varphi]\!]_X$ means by definition that $p \in (\nu_X \circ c)(x)$. Since the diagram commutes then $p \in (\nu_R \circ [c,d])(x,y) \Leftrightarrow p \in (\nu_Y \circ d)(y)$, that is, $y \in [\![\varphi]\!]_Y$.
2. Let us suppose $\varphi = \neg\varphi_0$. In this case we must show that $x \in \neg[\![\varphi_0]\!]_X$ if and only if $y \in \neg[\![\varphi_0]\!]_Y$, that is, we must see that $x \notin [\![\varphi_0]\!]_X$ if and only if $y \notin [\![\varphi_0]\!]_Y$. By induction hypothesis we have $x \in [\![\varphi_0]\!]_X$ if and only if $y \in [\![\varphi_0]\!]_Y$.
3. The cases $\varphi = \varphi_1 \wedge \varphi_2$ and $\varphi = \varphi_1 \vee \varphi_2$ are analogous to the previous case.
4. Let us suppose now that $\varphi = \bigcirc\varphi_0$. We must prove that $x \in \bigcirc[\![\varphi_0]\!]_X$ is equivalent to $y \in \bigcirc[\![\varphi_0]\!]_Y$, that is, $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$ is equivalent to $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$. The latter will be proved by structural induction on the functor F.

(a) $F = cnst$. In this case, both $\mathrm{Pred}(F)([\![\varphi_0]\!]_X)$ and $\mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$ are equal to $\top$, so we trivally get the result.

(b) $F = id$. In this case we must see that $c(x) \in [\![\varphi_0]\!]_X$ is equivalent to $d(y) \in [\![\varphi_0]\!]_Y$. Now, since we have xRy then, $c(x)Rd(y)$ and by induction hypothesis on φ, we know that, if aRb then $a \in [\![\varphi_0]\!]_X$ if and only if $b \in [\![\varphi_0]\!]_Y$.

(c) $F = F_1 \times F_2$. Let us suppose that $c(x) = (c_1(x), c_2(x))$ and $d(y) = (d_1(y), d_2(y))$. Then, if $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$ we have

$$c_1(x) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_X) \quad \text{and} \quad c_2(x) \in \mathrm{Pred}(F_2)([\![\varphi_0]\!]_X)\,.$$

By induction hypothesis on F_1 and F_2 we get $d_1(y) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_Y)$ and $d_2(y) \in \mathrm{Pred}(F_2)([\![\varphi_0]\!]_Y)$, so we get $y \in [\![\varphi]\!]_Y$. The other implication is analogous.

(d) $F = F_1 + F_2$. Let us suppose that $c(x) = \kappa_1(c_1(x)) = (c_1(x), 1)$; in this case we have $c_1(x) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_X)$. Let us define:

$$\begin{array}{rl} c_X : X \to F_1 X & \qquad d_Y : Y \to F_1 Y \\ z \;\mapsto c_1(x) & \qquad z \;\mapsto d_1(y) \end{array}$$

Trivially, R is a bisimulation between c_X and d_Y; then, if we apply the induction hypothesis on the functor we get $d_1(y) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_Y)$ and hence $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$. Analogously, if we suppose $c(x) = \kappa_2(c_1(x))$ we also get $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$.
The other implication is similar.

(e) $F = G^A$. Let us prove only one implication since the other one is almost identical. We have

$$\mathrm{Pred}(F)([\![\varphi_0]\!]_X) = \{f \mid \forall a \in A.\ \mathrm{Pred}(G)([\![\varphi_0]\!]_X)(f(a))\}\,.$$

Once again, as we have shown in other proofs, we define for each $a \in A$ and each F-coalgebra $c : X \longrightarrow F(X)$ a G-coalgebra, $c^a : X \longrightarrow G(X)$ where for each $x \in X$ we have $c^a(x) = c(x)(a)$. In this way, we have xRy and $c^a(x) = c(x)(a) \in \mathrm{Pred}(()G)([\![\varphi_0]\!]_X)$. By induction hypothesis we have that $d^a(y) \in \mathrm{Pred}(G)([\![\varphi_0]\!]_Y)$. Since this is a valid argument for all $a \in A$, we obtain $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$.

(f) $F = \mathcal{P}(G)$. Let us show only one of the implications. Let us suppose $d(y) \in \{U \mid \forall u \in U.\ \mathrm{Pred}(G)([\![\varphi_0]\!]_Y)(u)\}$; then, for all $b \in d(y)$, we have $b \in \mathrm{Pred}(G)([\![\varphi_0]\!]_Y)$. Let us show that $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$, or equivalently, that for all $a \in c(x)$, $a \in \mathrm{Pred}(G)([\![\varphi_0]\!]_X)$. Let us take an arbitrary $a \in c(x)$ and we define the constant coalgebra:

$$\begin{array}{rl} c^a_X : X \to GX \\ z \;\mapsto a \end{array}$$

Now, since xRy and R is a bisimulation, then $c(x)\mathrm{Rel}(F)(R)d(y)$, so there must exist some $b \in d(y)$ such that $a\mathrm{Rel}(G)(R)b$.

So we define:

$$\begin{aligned} d^b_Y : Y &\to GY \\ z &\mapsto b \end{aligned}$$

Trivially R is a bisimulation between the coalgebras c^a_X and d^b_Y and also $d^b_Y(y) = b \in \mathrm{Pred}(G)([\![\varphi_0]\!]_Y)$, hence by induction hypothesis it follows that $c^a_X(x) = a \in \mathrm{Pred}(G)([\![\varphi_0]\!]_X)$. This argument is valid for all $a \in c(x)$, therefore, $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$.

(g) $F = G^*$. Applying an analogous reasoning we get the result.

5. $\varphi = \Box\varphi_0$. Assuming that $x \in [\![\varphi]\!]_X$ we get that there exists

$$Q \subseteq X \text{ an invariant for } c \text{ with } Q \subseteq [\![\varphi_0]\!]_X \text{ and } x \in Q.$$

Now, RQ is a invariant for d and, also, such that $RQ \subseteq [\![\varphi_0]\!]_Y$ with $y \in RQ$. Indeed, if $x \in Q$ then $y \in RQ$ and if $b \in RQ$ there must exists some $a \in Q \subseteq [\![\varphi_0]\!]_X$ such that aRb. So, by induction hypothesis we get that $b \in [\![\varphi_0]\!]_Y$

On the other hand, if $y \in [\![\varphi]\!]_Y$ there must exists some invariant T on Y, such that $T \subseteq [\![\varphi_0]\!]_Y$ with $y \in T$, hence for proving $x \in [\![\varphi]\!]_X$ it is enough to consider the invariant $R^{-1}T$.

6. $\varphi = \Diamond\varphi_0$. We must prove $x \in \neg\Box\neg[\![\varphi_0]\!]_X$ if and only if $y \in \neg\Box\neg[\![\varphi_0]\!]_Y$. Or equivalently, $x \in \Box\neg[\![\varphi_0]\!]_X$ if and only if $y \in \Box\neg[\![\varphi_0]\!]_Y$. Let us show only one of the implications. If $y \in \Box\neg[\![\varphi_0]\!]_Y$ then, by definition, there exists an invariant $T \subseteq Y$ such that $T \subseteq \neg[\![\varphi_0]\!]_Y$ with $y \in T$. Once again, taking the invariant $R^{-1}T$ we get by induction hypothesis that $R^{-1}T \subseteq \neg[\![\varphi_0]\!]_X$ and $x \in R^{-1}T$, as required.

7. $\varphi = \varphi_1 \mathcal{U} \varphi_2$. We are going to proceed in a similar way as we did in Prop. 4. We must prove $y \in [\![\varphi_1]\!]_Y \,\mathcal{U}\, [\![\varphi_2]\!]_Y$ if and only if $x \in [\![\varphi_1]\!]_X \,\mathcal{U}\, [\![\varphi_2]\!]_X$.
The induction hypothesis give us the next property: if xRy then,

$$y \in [\![\varphi_i]\!]_Y \Leftrightarrow x \in [\![\varphi_i]\!]_X \quad \forall i \in \{1,2\}\,.$$

Hence, we have both $R^{-1}[\![\varphi_i]\!]_Y \subseteq [\![\varphi_i]\!]_X$ and $R[\![\varphi_i]\!]_X \subseteq [\![\varphi_i]\!]_Y$, for $i \in \{1,2\}$. This way, we can equivalently prove the following: $y \in [\![\varphi_1]\!]_Y \,\mathcal{U}\, [\![\varphi_2]\!]_Y$ implies $x \in R^{-1}[\![\varphi_1]\!]_Y \,\mathcal{U}\, R^{-1}[\![\varphi_2]\!]_Y$ and $x \in [\![\varphi_1]\!]_X \,\mathcal{U}\, [\![\varphi_2]\!]_X$ implies $y \in R[\![\varphi_1]\!]_X \,\mathcal{U}\, R[\![\varphi_2]\!]_X$.
Once again, we just show one of the implications. Let us show that

$$x \in [\![\varphi_1]\!]_X \,\mathcal{U}\, [\![\varphi_2]\!]_X \Longrightarrow y \in R[\![\varphi_1]\!]_Y \,\mathcal{U}\, R[\![\varphi_2]\!]_Y\,.$$

Also, we consider the auxiliary function

$$\begin{aligned} f^U_{(P,Q)} : \mathcal{P}(Y) &\longrightarrow \mathcal{P}(Y) \\ S &\longmapsto Q \cup (P \cap \neg \bigcirc \neg S)\,, \end{aligned}$$

with the notation

$$f_1 \quad \text{denotes} \quad f^U_{([\![\varphi_1]\!]_X, [\![\varphi_2]\!]_X)}(S)$$

$$f_2 \quad \text{denotes} \quad f^U_{(R[\![\varphi_1]\!]_Y, R[\![\varphi_2]\!]_Y)}(S)\,.$$

Recall that f_1 and f_2 satisfy $Rf_1(S) \subseteq f_2(RS)$ and that by Prop. 6 $f^U_{(P,Q)}$ is monotonic and $\cup$-continuous for any pair of predicates P and Q. Also, by Prop. 6 we have that the least fixed points are given by

$$\mu S.f_1(S) = \bigcup_{i=1}^{\infty} f_1^i(\emptyset)$$

$$\mu S.f_2(S) = \bigcup_{i=1}^{\infty} f_2^i(\emptyset)\,.$$

Hence, since $x \in \bigcup_i^\infty f_1^i(\emptyset)$ then, for some i we have $x \in f_1^i(\emptyset)$, so $y \in Rf_1^i(\emptyset)$. If we consider in Lemma 7 the bisimulation R^{-1} we get

$$y \in Rf_1^i(\emptyset) \subseteq f_2({Rf_1}^{i-1}(\emptyset))\,,$$

and by monotonicity of f_2,

$$f_2({Rf_1}^{i-1}(\emptyset)) \subseteq f_2(f_2({Rf_1}^{i-2}(\emptyset)))\,.$$

If we iterate this process we finally get

$$y \in {f_2}^i(R\emptyset) = {f_2}^i(\emptyset)\,.$$

And, that way, we get $y \in \bigcup_i^\infty f_2^i(\emptyset) = \mu S.f_2(S)$.

□

To obtain a similar result for simulations, we will need again to restrict the class of functors and orders as we did in Sections 4.1 and 4.2. In particular we are interested in the following antimonotonicity property: if $u \sqsubseteq u'$ then $\nu(u') \subseteq \nu(u)$.

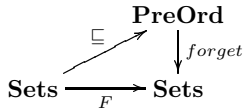
Definition 10. *Let $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ be a functor, AP a set of atomic propositions and $\nu : F \Rightarrow \mathcal{P}(AP)$ a natural transformation. We say that $\sqsubseteq$ is a down-natural ν-order if, whenever $u \sqsubseteq u'$ then $\nu(u') \subseteq \nu(u)$.*

Obviously this definition depends on the natural transformation that we consider in each case. For example, for Kripke structures we have the following natural transformation: $\nu_X((A_X, B_X)) = A_X \subseteq AP$. To obtain a down-natural ν-order the following must hold: $(u, v) \sqsubseteq (u', v')$ then $\nu((u', v')) \subseteq \nu((u, v))$, that is, it will be enough to require $(u, v) \sqsubseteq (u', v')$ iff $u' \subseteq u$.

This way, if we combine the down-closed and the down-natural orders we get:

$$\text{If} \quad (u, v) \sqsubseteq (u', v') \quad \text{then} \quad u' \subseteq u \text{ and } v \subseteq v'\,.$$

This characterization is not as restrictive as one could think. Indeed, if we recall the definition of functorial order we had:



This diagram means that the functor F and the order $\sqsubseteq$ almost have the same structure and indeed, we could use a natural transformation between $\sqsubseteq$ and $\mathcal{P}(AP)$ in Definition 9 instead of a natural transformation between F and $\mathcal{P}(AP)$, that is, $\nu :\sqsubseteq \Rightarrow \mathcal{P}(AP)$. Considering ν in this way, an immediate consequence is that if we take as order in $\mathcal{P}(AP)$ the relation $\supseteq$ (as is done in [15]), then $u \sqsubseteq v$ implies $\nu(u) \sqsubseteq \nu(v)$.

We can tackle the proof of reflection of properties (with atomic propositions) by simulations as we did in Section 4.1, imposing to the order not only to be down-natural but also down-closed. But, if we do that we will find the same difficulties we faced in Section 4.1 (that is, we would not be able to prove reflection of formulas built with the operators *until* and *eventually*). Therefore, we must restrict the class of functors and orders, as we did with the class **Order** in Section 4.2, but imposing also that the orders must be down-natural.

Definition 11. *The class* **Down-Natural ν-Order** *is the subclass of* **Order** *where all orders are down-natural.*

Notice that we are defining a different class for each natural transformation ν.

Under this condition we state and prove the corresponding theorem involving simulations and the reflection of properties (with atomic propositions).

Theorem 6. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ on the same polynomial functor F from* **Sets** *to* **Sets** *belonging to the class* **Down-Natural ν-Order** *and let φ be a temporal formula. Then, for each $x \in X$ and $y \in Y$ such that xRy:*

$$y \in [\![\varphi]\!]_Y \quad \Longrightarrow \quad x \in [\![\varphi]\!]_X \, .$$

Proof. We will prove the theorem by structural induction on φ.

1. Let $\varphi = p$ where p is an atomic proposition. Let us suppose that $y \in [\![p]\!]_Y$, so $p \in (\nu_Y \circ d)(y)$. Since R is a simulation there must exists u and v such that $c(x) \sqsubseteq u\mathrm{Rel}(F)(R)v \sqsubseteq d(y)$. We must prove that $x \models p$, that is, $p \in \nu_X(c(x))$. Since we are supposing that $z \sqsubseteq z'$ implies $\nu(z') \subseteq \nu(z)$, we have that $p \in \nu_Y(v)$; so it will be enough to show that $p \in \nu_X(u)$ and use that we are dealing with a down-natural order.
   Let us show that $\nu_X(u) = \nu_Y(v)$. By definition, $\mathrm{Rel}(F)(R)$ is the image of $\langle Fr_1, Fr_2\rangle : FR \longrightarrow FX \times FY$. Hence, since $u\mathrm{Rel}(F)(R)v$, there exists some $w \in FR$ such that $(u, v) = (Fr_1(w), Fr_2(w))$; so $p \in \nu_Y(v) = \nu_R(w) = \nu_X(u)$, as the following diagram shows:

$$\begin{array}{ccccc} FX & \xleftarrow{Fr_1} & FR & \xrightarrow{Fr_2} & FY \\ \downarrow{\scriptstyle \nu_X} & & \downarrow{\scriptstyle \nu_R} & & \downarrow{\scriptstyle \nu_Y} \\ \mathcal{P}(AP) & \xleftarrow{id} & \mathcal{P}(AP) & \xrightarrow{id} & \mathcal{P}(AP) \end{array}$$

2. The cases $\varphi = \varphi_1 \wedge \varphi_2$ are $\varphi = \varphi_1 \vee \varphi_2$ trivial.
3. Let us suppose that $\varphi = \bigcirc\varphi_0$. We must prove that $y \in \bigcirc[\![\varphi_0]\!]_Y$ implies $x \in \bigcirc[\![\varphi_0]\!]_X$, that is, that $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$ implies $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$. Once again this will be proved applying structural induction over the functor F.
   (a) $F = cnst$. In this case both $\mathrm{Pred}(F)([\![\varphi_0]\!]_Y) = \top$ and $\mathrm{Pred}(F)([\![\varphi_0]\!]_X) = \top$, so we trivially get the result.
   (b) $F = id$. In this case we have that both $\sqsubseteq$ and $\mathrm{Pred}(F)$ are the identity. Hence, we must show that $c(x) \in [\![\varphi_0]\!]_X$. Since R is a simulation we have $c(x) \sqsubseteq uRv \sqsubseteq d(y)$, that is, $c(x)Rd(y)$. By induction hypothesis over φ we have that, if xRy, $y \in [\![\varphi_0]\!]_Y$ implies $x \in [\![\varphi_0]\!]_X$, and that way we obtain $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$.
   (c) Let F_1 and F_2 be two functors with down-natural orders $\sqsubseteq^1$ and $\sqsubseteq^2$ and let us consider $F = F_1 \times F_2$ with the down-closed order given by Def. 7. In this case we have

$$\mathrm{Pred}(F)(P) = \{(u, v) \mid \mathrm{Pred}(F_1)(P)(u) \wedge \mathrm{Pred}(F_2)(P)(v)\}\,.$$

   Let us suppose $d(y) = (d_1(y), d_2(y))$ and $c(x) = (c_1(x), c_2(x))$. Henceforth, if $d(y) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$ then $d_1(y) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_Y)$ and $d_2(y) \in \mathrm{Pred}(F_2)([\![\varphi_0]\!]_Y)$. Now, since R is a simulation between c and d then, given xRy, it follows the existence of $u = (u_1, u_2)$ and $v = (v_1, v_2)$ such that $c(x) \sqsubseteq (u_1, u_2)\mathrm{Rel}(F)(R)(v_1, v_2) \sqsubseteq d(y)$. By definition of $\sqsubseteq$ we have $c_1(x) \sqsubseteq^1 u_1\mathrm{Rel}(F_1)(R)v_1 \sqsubseteq^1 d_1(y)$ and $c_2(x) \sqsubseteq^2 u_2\mathrm{Rel}(F_2)(R)v_2 \sqsubseteq^2 d_2(y)$. That is, R is also a simulation between c_1 and d_1, and c_2 and d_2. Since both orders $\sqsubseteq^1$ and $\sqsubseteq^2$ are down-natural, we can apply the induction hypothesis over F and obtain that $c_1(x) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_X)$ and $c_2(x) \in \mathrm{Pred}(F_2)([\![\varphi_0]\!]_X)$, hence we obtain $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$, as we wanted to prove.
   (d) Let F_1 and F_2 be two functors with down-natural orders $\sqsubseteq^1$ and $\sqsubseteq^2$ and let us consider $F = F_1 + F_2$ with the order given by Definition 7.
   Without loss of generality let us suppose that $d(y) = \kappa_1(d_1(y)) = (d_1(y), 1)$; we have $d_1(y) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_Y)$. Let us consider now the following constant coalgebras:

$$\begin{array}{rl} c_X : X \to F_1X & \qquad d_Y : Y \to F_1Y \\ z \mapsto c_1(x) & \qquad z \mapsto d_1(y) \end{array}$$

   Since R is a simulation and the order is the disjoint sum, we also have that R is a simulation between c_X and d_Y with down-natural orders; hence, by induction hypothesis we have $c_1(x) \in \mathrm{Pred}(F_1)([\![\varphi_0]\!]_X)$ and in this way, $c(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$.
   (e) Let F be a functor with a down-natural order $\sqsubseteq^F$ and let us consider the functor F^A with the order given by Def. 7. Since R is a simulation, there exists u and v such that $c(x) \sqsubseteq u\mathrm{Rel}(F^A)(R)v \sqsubseteq d(y)$. Now, for each $a \in A$ and each F^A-coalgebra $d : Y \longrightarrow F^A(Y)$ we define a coalgebra over F this way: $d^a : Y \longrightarrow F(Y)$ where, for each $y \in Y$, $d^a(y) = d(y)(a)$;

analogously we define $c^a(x) = c(x)(a)$ for all $x \in X$. In this way we have xRy and $d^a(y) = d(y)(a) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$.
Now, by definition of $\sqsubseteq$ we have $c^a(x) \sqsubseteq^F u(a)\ \mathrm{Rel}(F)(R)v(a) \sqsubseteq^F d^a(y)$, that is, R is a simulation between c^a and d^a. By induction hypothesis we have $c^a(x) \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$. Since this argument is valid for all $a \in A$, we have that $c(x) \in \mathrm{Pred}(F^A)([\![\varphi_0]\!]_X)$.

(f) Let F be a functor with a down-natural order $\sqsubseteq^F$ and let us consider the functor $\mathcal{P}(F)$ with the order given by Def. 7. In this case we have $d(y) \in \mathrm{Pred}(\mathcal{P}(F))([\![\varphi_0]\!]_Y)$, so for all $b \in d(y)$ we will have that $b \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$, and we have to prove that $c(x) \in \mathrm{Pred}(\mathcal{P}(F))([\![\varphi_0]\!]_X)$, or equivalently, that for all $a \in c(x)$ we have $a \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$. Let us take an arbitrary $a \in c(x)$ and define the following constant coalgebra:

$$\begin{aligned} c^a_X : X &\to FX \\ z &\mapsto a \end{aligned}$$

Since xRy and R is a simulation we have $c(x) \sqsubseteq u\mathrm{Rel}(\mathcal{P}(F))(R)v \sqsubseteq d(y)$. By definition of $\sqsubseteq$, it follows that since $c(x) \sqsubseteq u$, then for each $a \in c(x)$ there exists $a_1 \in u$ such that $a \sqsubseteq^F a_1$. Also, by the definition of relation lifting it follows that for each element $a_1 \in u$ there exists an element $b_1 \in v$ such that $a_1 \mathrm{Rel}(F)(R) b_1$. Again, by the definition of the order it follows that for each $b_1 \in V$ there exists a $b \in d(y)$ such that $b_1 \sqsubseteq^F b$.
Now, we define the following:

$$\begin{aligned} d^b_Y : Y &\to FY \\ z &\mapsto b \end{aligned}$$

Trivially, R is a simulation between c^a_X and d^b_Y, and also $d^b_Y = b \in \mathrm{Pred}(F)([\![\varphi_0]\!]_Y)$; hence by induction hypothesis over F it follows $c^a_X = a \in \mathrm{Pred}(F)([\![\varphi_0]\!]_X)$. Since this is a valid argument for each $a \in c(x)$, it follows that $c(x) \in \mathrm{Pred}(\mathcal{P}(F))([\![\varphi_0]\!]_X)$.

4. $\varphi = \Box\varphi_0$. Suppose that $y \in [\![\varphi]\!]_Y$: there exists an invariant $Q \subseteq Y$ such that $Q \subseteq [\![\varphi_0]\!]_Y$ and $y \in Q$. Now, recall that the functors of the class **Order** satisfied Lemmas 10 and 11; hence, $R^{-1}Q$ is an invariant too. Furthermore $R^{-1}Q \subseteq [\![\varphi_0]\!]_X$ with $x \in R^{-1}Q$. Indeed, since $y \in Q$ then $x \in R^{-1}Q$; on the other hand, if $a \in R^{-1}Q$ there must exist some $b \in Q \subseteq [\![\varphi_0]\!]_Y$ such that aRb. Hence, by induction hypothesis, $a \in [\![\varphi_0]\!]_X$ so $x \in [\![\varphi_0]\!]_X$ as requested.
5. $\varphi = \Diamond\varphi_0$. We must prove that $y \in \neg\Box\neg[\![\varphi_0]\!]_Y$ implies $x \in \neg\Box\neg[\![\varphi_0]\!]_X$, or equivalently, that $x \in \Box\neg[\![\varphi_0]\!]_X$ implies $y \in \Box\neg[\![\varphi_0]\!]_Y$. Indeed, as in the previous case there exists an invariant $T \subseteq X$ such that $T \subseteq \neg[\![\varphi_0]\!]_X$ with $x \in T$. Once again, RT is an invariant such that $RT \subseteq Y$, $RT \subseteq \neg[\![\varphi_0]\!]_Y$ and $y \in RT$, as required.
6. $\varphi = \varphi_1 \mathcal{U} \varphi_2$. We are going to prove that $y \in [\![\varphi_1]\!]_Y\ \mathcal{U}\ [\![\varphi_2]\!]_Y$ implies $x \in [\![\varphi_1]\!]_X\ \mathcal{U}\ [\![\varphi_2]\!]_X$.
As we showed in the proof of Theorem 5, the induction hypothesis provides us with the following property: if xRy then

$$y \in [\![\varphi_i]\!]_Y \Rightarrow x \in [\![\varphi_i]\!]_X \quad \forall i \in \{1, 2\}\,.$$

Hence, we have that $R^{-1}[\![\varphi_i]\!]_Y \subseteq [\![\varphi_i]\!]_X$, for $i \in \{1,2\}$. So we must prove that $y \in [\![\varphi_1]\!]_Y \,\mathcal{U}\, [\![\varphi_2]\!]_Y$ implies $x \in R^{-1}[\![\varphi_1]\!]_Y \,\mathcal{U}\, R^{-1}[\![\varphi_2]\!]_Y$.
Once again we define

$$\begin{array}{rcl} f^U_{(P,Q)} : \mathcal{P}(Y) & \longrightarrow & \mathcal{P}(Y) \\ S & \longmapsto & Q \cup (P \cap \neg \bigcirc \neg S)\,, \end{array}$$

with the following notation:

$$f_1 \quad \text{denotes} \quad f^U_{([\![\varphi_1]\!]_Y, [\![\varphi_2]\!]_Y)}(S)$$

$$f_2 \quad \text{denotes} \quad f^U_{(R^{-1}[\![\varphi_1]\!]_Y, R^{-1}[\![\varphi_2]\!]_Y)}(S)\,.$$

Recall that since Lemmas 11 and 10 are satisfied, we can guarantee that f_1 and f_2 satisfy the relation $R^{-1}f_1(S) \subseteq f_2(R^{-1}S)$. On the other hand, since $f^U_{(P,Q)}$ is monotonic and $\cup$-continuos we have that

$$\mu S.f_1(S) = \bigcup_{i=1}^{\infty} f_1^i(\emptyset)$$

$$\mu S.f_2(S) = \bigcup_{i=1}^{\infty} f_2^i(\emptyset)\,.$$

Hence, since $y \in \bigcup_i^\infty f_1^i(\emptyset)$ then, for some i we have $y \in f_1^i(\emptyset)$, so, $y \in Rf_1^i(\emptyset)$, and also

$$x \in R^{-1}f_1^i(\emptyset) \subseteq f_2(R^{-1}{f_1}^{i-1}(\emptyset))\,.$$

By monotonicity of f_2,

$$f_2(R^{-1}{f_1}^{i-1}(\emptyset)) \subseteq f_2(f_2(R^{-1}{f_1}^{i-2}(\emptyset)))\,.$$

If we iterate this process we finally get

$$x \in {f_2}^i(R\emptyset) = {f_2}^i(\emptyset)\,.$$

And that way $x \in \bigcup_i^\infty f_2^i(\emptyset) = \mu S.f_2(S)$, as required.

□

We showed above that simulations for functors in the class **Order** reflected and preserved all kinds of properties. Instead, now we can only prove one implication, that corresponding to the reflection of properties. This is so because down-natural ν-orders have a natural direction.

Exactly in the same way as we did with down-natural ν-orders, we can define the corresponding class of up-natural ν-orders:

Definition 12. *Let $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ be a functor, AP a set of atomic propositions and $\nu : F \Rightarrow \mathcal{P}(AP)$ a natural transformation. We say that $\sqsubseteq$ is an up-natural ν-order if $u \sqsubseteq u'$ implies $\nu(u) \subseteq \nu(u')$.*

Also, as we did for down-natural ν-orders, we must define a subclass of **Order**:

Definition 13. *The class* **Up-Natural ν-Order** *is the subclass of* **Order** *where all orders are up-natural.*

Theorem 7. *Let R be a simulation between coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ on the same polynomial functor F in the class* **Up-Natural ν-Order**, *and let φ be a temporal formula. Then, for all $x \in X$ and $y \in Y$ such that xRy:*

$$x \in [\![\varphi]\!]_X \quad \Longrightarrow \quad y \in [\![\varphi]\!]_Y \,.$$

6 Conclusions

The main goal of this paper was to study under what assumptions coalgebraic simulations reflect properties. In our way towards the proof of this result, we were also able to prove reflection and preservation of properties by coalgebraic bisimulations. For expressing the properties we used Jacobs' temporal logic [8], later extended with atomic propositions using the idea presented in [12].

That coalgebraic bisimulations reflect and preserve properties expressed in modal logic is a well-known topic (e.g, [9,12,16]), but not so the corresponding results for simulations. The main difficulty is that Hughes and Jacobs' notion of simulation is defined by means of an arbitrary functorial order which bestows them with a high degree of freedom. We have dealt with this by restricting the class of functorial orders (although even so we are not able of obtaining a general result) and by restricting also the class of allowed functors.

In order to get more general results on the subject, an interesting path that we intend to explore is the search for a canonical notion of simulation. This definition would provide us, not only with a "natural" way to understand simulations but, hopefully, would also give rise to "natural" general results about reflection of properties.

Another promising direction of research is the study of reflection and preservation of properties in probabilistic systems following our results of [4] in combination with the ideas presented in [6,5,2].

References

1. P. Aczel and N. P. Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, volume 389, pages 357–365, 1989.
2. F. Bartels, A. Sokolova, and E.P. de Vink. A hierarchy of probabilistic system types. *Theor. Comput. Sci.*, 327(1-2):3–22, 2004.
3. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
4. D. de Frutos Escrig, M. Palomino, and I. Fábregas. Multiset bisimulation as a common framework for ordinary and probabilistic bisimulations. Submitted, 2007.

5. E.P. de Vink and J.J.M.M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. In P. Degano et al, editors, *ICALP'97*, volume 1256 of *Lect. Notes Comp. Sci.*, pages 4460–470. Springer, Berlin, 1997.
6. I. Hasuo. Generic forward and backward simulations. In *International Conference on Concurrency Theory (CONCUR 2006)*, volume 4137 of *Lect. Notes Comp. Sci.*, pages 406–420. Springer, 2006.
7. J. Hughes and B. Jacobs. Simulations in coalgebra. *Theor. Comput. Sci.*, 327(1-2):71–108, 2004.
8. B. Jacobs. *Introduction to Coalgebra. Towards Mathematics of States and Observations*. Book in preparation. Draft available in the web.
`http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`
9. B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1999.
10. B. Jacobs and J.J.M.M. Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 62:222–259, 1997.
11. Y. Kesten and A. Pnueli. Control and data abstraction: The cornerstones of practical formal verification. *International Journal on Software Tools for Technology Transfer*, 4(2):328–342, 2000.
12. A. Kurz. *Logics for coalgebras and applications to computer science*. PhD thesis, Universität München, 2000.
13. C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6:1–36, 1995.
14. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
15. M. Palomino. *Reflexión, abstracción y simulación en la lógica de reescritura*. PhD thesis, Universidad Complutense de Madrid, Spain, March 2005.
16. D. Pattinson. *Expressivity Results in the Modal Logic of Coalgebras*. PhD thesis, Universität München, 2001.
17. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

Non-Strongly Stable Orders Also Define Interesting Simulation Relations*

Ignacio Fábregas, David de Frutos Escrig, and Miguel Palomino

Departamento de Sistemas Informáticos y Computación, UCM
fabregas@fdi.ucm.es {miguelpt, defrutos}@sip.ucm.es

Abstract. We present a study of the notion of coalgebraic simulation introduced by Hughes and Jacobs. Although in their original paper they allow any functorial order in their definition of coalgebraic simulation, for the simulation relations to have good properties they focus their attention on functors with orders which are strongly stable. This guarantees a so-called "composition-preserving" property from which all the desired good properties follow. We have noticed that the notion of strong stability not only ensures such good properties but also "distinguishes the direction" of the simulation. For example, the classic notion of simulation for labeled transition systems, the relation "p is simulated by q", can be defined as a coalgebraic simulation relation by means of a strongly stable order, whereas the opposite relation, "p simulates q", cannot. Our study was motivated by some interesting classes of simulations that illustrate the application of these results: covariant-contravariant simulations and conformance simulations.

1 Introduction and presentation of our new results

Simulations are a very natural way to compare systems defined by transition systems or other related mechanisms based on the description of systems by means of the actions they can execute at each of their states [11]. They can be enriched in several ways to obtain, in particular, the important ready simulation semantics [2,8], as well as other more elaborated ones such as nested simulations [5]. Quite recently we have studied the general concept of constrained simulation [3], proving that all the simulation relations constrained by an adequate condition have similar properties. The semantics of these constrained simulations is also the basis for our unified presentation of the semantics of processes [4], where all the semantics in the ltbt-spectrum [13] (and other new semantics) are classified in a systematic way.

Hughes and Jacobs [6] have also developed a systematic study of simulation-like relations, this time in a purely coalgebraic context, so that simulations are studied in connection with bisimulations [11], the fundamental concept to define

* Research supported by the Spanish projects DESAFIOS TIN2006-15660-C02-01, WEST TIN2006-15578-C02-01, PROMESAS S-0505/TIC/0407 and UCM-BSCH GR58/08/910606.

equivalence in the coalgebraic world. Their coalgebraic simulations are defined in terms of an order $\sqsubseteq$ associated to the functor F corresponding to the coalgebra $c : X \longrightarrow FX$ that we want to observe. In this way they obtain a very general notion of coalgebraic simulation, not only because all functors F are considered, including in particular the important class of polynomial functors, but also because by changing the family of orders $\sqsubseteq_X$ many different families of simulation relations can be obtained. The general properties of these simulations can be studied in the defined coalgebraic framework, thus avoiding the need of similar proofs for each of the particular classes of simulations.

Certainly, this generic presentation of the notion of coalgebraic simulation has as advantage that it provides a wide and abstract framework where one can try to isolate and take advantage of the main properties of all the simulation-like relations. However, at the same time it can be argued that the proposal fails to capture in a tight manner the spirit of simulation relations because, in addition to the natural notions of simulations, the framework also allows for other less interesting relations. This has as a result that some natural properties of simulations cannot be proved in general, simply due to the fact that they are not satisfied by all of the permitted coalgebraic simulation relations. For instance, the induced similarity relation between systems is not always an order because transitivity is not always satisfied. In order to guarantee transivity, and other related properties of coalgebraic simulations, Jacobs and Hughes introduce in [7] the composition-preserving property to the order $\sqsubseteq$ that induces the simulation relation. In [6] they continue with the study of the topic and present *stability* of orders as a natural categorical property to guarantee that an order is composition-preserving. They also comment that stability is not easy to check and introduce a stronger condition (that we will call right-stability) so that, whenever applicable, the checking of the main properties of coalgebraic simulations becomes much simpler than in the general case.

Roughly speaking, given an order $\sqsubseteq_X$ on FX for each set X, the induced coalgebraic simulations are defined in the same way as bisimulations for F, but allowing a double application of $\sqsubseteq$ on the two sides of the defined relation. More precisely, instead of the functor $\mathrm{Rel}(F)$ defining plain bisimulations, $\mathrm{Rel}_{\sqsubseteq}(F)$ defined as $\sqsubseteq_Y \circ \mathrm{Rel}(F) \circ \sqsubseteq_X$ is used. There are several interesting facts hidden behind the apparent simplicity of this definition. The first one is that, in general, it only defines an order and not an equivalence relation, even if it is based on bisimulations (that always define an equivalence relation, namely, bisimilarity). The reason is that the order $\sqsubseteq$ appears "in the same direction" on both sides of the definition, thus breaking its symmetry. However, we can also define some equivalence relations weaker than bisimilarity by using an equivalence relation $\equiv$ as the order $\sqsubseteq$. Another interesting fact is that whenever we define a coalgebraic simulation by using $\sqsubseteq$, the inverse order $\sqsupseteq$ defines the inverse relation of that defined by $\sqsubseteq$ once we also interchange the roles of the related sets X and Y (so we could say that we are defining in fact the same relation but looking at it from the other side). Stability is also a symmetric condition, so that whenever an order $\sqsubseteq$ on a functor F is stable, the inverse order $\sqsupseteq$ is stable for F, too. This

is quite reasonable, since stability is imposed in order to guaratee transitivity of the generated similarity relation and the inverse of a transitive relation is also transitive, so that whenever $\sqsubseteq$ generates an "admissible" similarity relation (meaning that it is an order), the inverse order $\sqsupseteq$ must be also admissible.

It is worth noting that the stronger condition guaranteeing stability is asymmetric. In fact, Hughes and Jacobs prove in [6] that "right-stability" implies that

$$\mathrm{Rel}(F)(R) \circ \sqsubseteq_X \subseteq \sqsubseteq_Y \circ \mathrm{Rel}(F)(R), \tag{1}$$

which in fact motivates our name for the condition.

A second surprise was to notice that, in most cases, right-stability induces a "natural direction" on the orders defining the coalgebraic simulation. For instance, for plain similarity over labeled transition systems, the inclusion order $\subseteq$ induces the classic simulation relation while the reversed inclusion $\supseteq$ induces the opposite "simulated by" relation: the first one is right-stable while the second is not.

All these general results arose when trying to integrate two new simulation-like notions as coalgebraic simulations definable by a stable order, so that we could obtain for free all the good properties that have been proved in [6] for this class of relations.

The first new simulation notion is that of covariant-contravariant simulations, where the alphabet of actions Act is partitioned into three disjoint sets Act^l, Act^r, and Act^{bi}. The intention is for the simulation to treat the actions in Act^l like in the ordinary case, to interchange the role of the related processes for those actions in Act^r, and to impose a symmetric condition like that defining bisimulation for the actions in Act^{bi}.

The second notion, conformance simulations, captures the conformance relations [9,12] that several authors introduced in order to formalize the notion of possible implementations. Like covariant-contravariant simulations, they can be defined as coalgebraic simulations for some stable order which is not right-stable neither left-stable. We show that the good properties of these two classes of orders are preserved in those orders that can be seen as a kind of composition of right-stable and left-stable orders. We use this fact to derive the stability of the orders defining both covariant-contravariant and conformance simulations.

2 Coalgebraic simulations and stability

Given a category $\mathbb{C}$ and an endofunctor F in $\mathbb{C}$, an F-coalgebra, or just a coalgebra, consists of an object $X \in \mathbb{C}$ together with a morphism $c : X \longrightarrow FX$. We often call X the state space and c the transition or coalgebra structure.

For example, labeled transition systems are coalgebras for the functor $F = \mathcal{P}(id)^A$, where A is the set of labels. Sometimes we will denote $\mathcal{P}^A$ for $\mathcal{P}(id)^A$.

An arbitrary endofunctor $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ can be lifted to a functor in the category $\mathbf{Rel}$ over $\mathbf{Sets} \times \mathbf{Sets}$ of relations, $\mathrm{Rel}(F) : \mathbf{Rel} \longrightarrow \mathbf{Rel}$. In set-theoretic terms, for a relation $R \subseteq X_1 \times X_2$,

$$\mathrm{Rel}(F)(R) = \{\langle u, v\rangle \in FX_1 \times FX_2 \mid \exists w \in F(R).\, F(r_1)(w) = u, F(r_2)(w) = v\}.$$

For example, in the case of labeled transition systems we have that

$$\mathrm{Rel}(\mathcal{P}(id)^A)(R) = \{(f,g) \mid \forall a \in A.\, (f(a), g(a)) \in \{(U,V) \mid \forall u \in U.\, \exists v \in V.\, uRv \wedge \forall v \in V.\, \exists u \in U.\, uRv\}\}$$

A *bisimulation* for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ which is "closed under c and d":

$$\text{if } (x,y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)(R),$$

where the r_i are the projections of R into X and Y. Sometimes we shall use the term F-bisimulation to emphasize the functor we are working with.

Bisimulations can also be characterized by means of spans, using the general categorical definition by Aczel and Mendler [1]:

$$\begin{array}{ccccc} X & \xleftarrow{r_1} & R & \xrightarrow{r_2} & Y \\ {\scriptstyle c}\downarrow & & {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle d} \\ FX & \xleftarrow{Fr_1} & FR & \xrightarrow{Fr_2} & FY \end{array}$$

R is a bisimulation iff it is the carrier of some coalgebra e making the above diagram commute. Alternatively, bisimulations can also be defined as the $\mathrm{Rel}(F)$-coalgebras in the category **Rel**.

We will also need the general concept of simulation introduced by Hughes and Jacobs [6] using orders on functors. Let $F : \mathbf{Sets} \longrightarrow \mathbf{Sets}$ be a functor. *An order on F* is defined by means of a functorial collection of preorders $\sqsubseteq_X \subseteq FX \times FX$ that must be preserved by renaming: for every $f : X \longrightarrow Y$, if $u \sqsubseteq_X u'$ then $Ff(u) \sqsubseteq_Y Ff(u')$.

Given an order $\sqsubseteq$ on F, a *$\sqsubseteq$-simulation* for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ such that

$$\text{if } (x,y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}_{\sqsubseteq}(F)(R),$$

where the lax relation lifting $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ is $\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X$, which can be expanded to

$$\mathrm{Rel}_{\sqsubseteq}(F)(R) = \{(u,v) \mid \exists w \in F(R).\, u \sqsubseteq_X Fr_1(w) \wedge Fr_2(w) \sqsubseteq_Y v\}.$$

Alternatively, $\sqsubseteq$-simulations are just the $\mathrm{Rel}_{\sqsubseteq}(F)$-coalgebras in **Rel**.

Sometimes, when $f : X \longrightarrow Y$ and $A \subseteq X$ we will simply write $f(A)$ for the image $\coprod_f(A)$.

A functor with order $\sqsubseteq$ is *stable* [6] if the relation lifting $\mathrm{Rel}_{\sqsubseteq}(F)$ commutes with substitution, that is, if for every $f : X \longrightarrow Z$ and $g : Y \longrightarrow W$, $\mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R)) = (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$.[1] They also define a stronger condition that we are going to call right-stability.

[1] In fact, the inclusion $\subseteq$ always holds.

Definition 1 ([6]). *We will say that a functor F with order $\sqsubseteq$ is **right-stable** if, for every function $f : X \longrightarrow Y$, we have*[2]

$$(id \times Ff)^{-1} \sqsubseteq_Y \ \subseteq \ \coprod_{Ff \times id} \sqsubseteq_X . \tag{2}$$

According to [6], condition (2) is equivalent to (a) F being stable and (b) for every relation $R \subseteq X \times Y$,

$$\mathrm{Rel}(F)(R) \circ \sqsubseteq_X \ \subseteq \ \sqsubseteq_Y \circ \mathrm{Rel}(F)(R). \tag{3}$$

Right-stability was introduced by arguing that it is easier to check than plain stability, while being satisfied by nearly all orders discussed in that paper. Surprisingly, one cannot find in [6] a clear explanation of the reason why right-stable orders are easier to analyze. In our opinion, the crucial fact is that from (3) we can immediately conclude that

$$\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X \ = \ \sqsubseteq_Y \circ \mathrm{Rel}(F)(R), \tag{4}$$

so that the coalgebraic simulations for a right-stable order $\sqsubseteq$ can be equivalently defined by means of the asymmetric definition on the right-hand side of equality (4). If the order $\sqsubseteq$ can be used only on one of the sides of the definition, the verification of the properties of the induced coalgebraic simulations becomes much easier than when using the original definition.

It was quite surprising to discover that the easiest way to prove the properties of the "simulated by" relations which come from symmetric properties such as composition-preserving (that are also satisfied by the corresponding inverse relations "simulates") is to break that symmetry by considering the asymmetric definition of coalgebraic simulations that only use $\sqsubseteq_Y$; certainly, this is only possible when the defining order $\sqsubseteq$ is right-stable.

Stability is used in [6, Lemma 5.3] to prove that lax relation lifting preserves composition of relations, which is needed to prove [6, Lemma 5.4(2)], the crucial fact that the induced similarity relation is transitive; this need not be the case for the simulation notion defined by an arbitrary order $\sqsubseteq$.

3 On stability of simulation and anti-simulation

Plain simulations between labeled transition systems can be defined as coalgebraic simulations considering the functor $F = \mathcal{P}^A$ (G^A denote the funtor $X \mapsto (G(X))^A$) with the order $\sqsubseteq$ given by $\alpha \sqsubseteq \beta$ for $\alpha, \beta : A \longrightarrow \mathcal{P}X$ iff for all $a \in A$, $\alpha(a) \subseteq \beta(a)$.

Lemma 1. *The order $\sqsubseteq$ defining plain simulations for labeled transition systems is right-stable.*

[2] Again, the other inclusion is always true since $\sqsubseteq$ functorial means that $Ff(u) \sqsubseteq_Y Ff(v)$ if $u \sqsubseteq_X v$.

Proof. We have to prove that given any $f : X \longrightarrow Y$ condition (2) holds. Let us take $a \in A$, and any pair $(y_a, x_a) \in (id \times \mathcal{P}f)^{-1} \subseteq_Y$, that is, $(y_a, x_a) \in \{(B, A) \in \mathcal{P}Y \times \mathcal{P}X \mid B \subseteq f(A)\}$. We have to prove that $(y_a, x_a) \in \coprod_{\mathcal{P}f \times id} \subseteq_X$, that is, if $(y_a, x_a) \in \{(f(A'), A) \in \mathcal{P}Y \times \mathcal{P}X \mid A' \subseteq A\}$, or equivalently, if there is some $x'_a \in \mathcal{P}X$ such that $x'_a \subseteq x_a$ and $f(x'_a) = y_a$. But, since $(y_a, x_a) \in (id \times \mathcal{P}f)^{-1} \subseteq_Y$ we have that $y_a \subseteq f(x_a)$, that is, there must exist the inverse image of y_a, that is, there is some $x'_a \in \mathcal{P}X$ such that $f(x'_a) = y_a$. But, since the direct image of a set over a function is monotonic and $f(x'_a) \subseteq f(x_a)$ then, also, $x'_a \subseteq x_a$, as we had to prove. □

Corollary 1. *Plain simulations between labeled transition systems can be defined as the* $(\sqsubseteq_Y \circ \mathrm{Rel}(F))$*-coalgebras.*

It is worth examining the consequences of the removal of $\sqsubseteq_X$ from the original definition of coalgebraic simulations in this particular case. Both $\sqsubseteq_X$ and $\sqsubseteq_Y$ correspond to the inclusion order, but when applied at the right-hand side it means that we can reduce the set of successors of the simulating process q when simulating the execution of a by p. This means that starting from a set $Y' \subseteq Y$ we can obtain an adequate subset $Y'' \subseteq Y'$. Instead, the application of $\sqsubseteq_X$ at the left-hand side allows to enlarge the set of successors of the simulated process p and this produces a set X'' larger than the given X': one could say that we need to consider "new" information not in X', while going from Y' to Y'' just "removes" some known information.

Another interesting point arises from the fact that every use of $\sqsubseteq_X$ at the left-hand side can be "compensated" by removing at Y the added states and this is why Corollary 1 was correct, because we can always avoid the introduction of new successors in the simulated process by simply removing them at the right-hand side. However, the opposite procedure, to compensate the removal of states by adding them at the simulated process side is not always possible, since in general X could be not big enough.

The anti-simulations can be defined as coalgebraic simulations by taking the reversed inclusion order instead of $\subseteq$. It is interesting to note that it is not right-stable as the following counterexample shows. Let $X = \{x\}$ and $Y = \{y_1, y_2\}$ and let $f : X \longrightarrow Y$ be such that $f(x) = y_1$. With these definitions the pair $(Y, X) \in (id \times \mathcal{P}f)^{-1}(\supseteq)$, since $Y \supseteq \{y_1\} = \mathcal{P}f(X)$, but it is obvious that there is no $A \subseteq X$ such that $Y = f(A)$ because f is not surjective.

However, the order defining anti-simulations is stable as a consequence of the following general result.

Lemma 2. *F with an order $\sqsubseteq$ is stable iff it is stable with the order $\sqsubseteq^{op}$.*

Proof. It is shown in [6, Lemma 4.2(4)] that $\mathrm{Rel}_{\sqsubseteq^{op}}(F)(R) = (\mathrm{Rel}_{\sqsubseteq}(F)(R^{op}))^{op}$. Then, on the one hand,

$$\begin{aligned}(Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq^{op}}(F)(R)) &= (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R^{op}))^{op} \\ &= ((Fg \times Ff)^{-1}\mathrm{Rel}_{\sqsubseteq}(F)(R^{op}))^{op},\end{aligned}$$

and on the other hand,

$$\begin{aligned}\mathrm{Rel}_{\sqsubseteq^{op}}(F)((f\times g)^{-1}(R)) &= (\mathrm{Rel}_{\sqsubseteq}(F)((f\times g)^{-1}(R))^{op})^{op}\\ &= (\mathrm{Rel}_{\sqsubseteq}(F)((g\times f)^{-1}(R^{op})))^{op}.\end{aligned}$$

Since $R^{op}\subseteq Y\times X$ is a relation whenever $R\subseteq X\times Y$ is so, and f, g, and R are arbitrary, we have shown that

$$\mathrm{Rel}_{\sqsubseteq}(F)((f\times g)^{-1}(R)) = (Ff\times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$$

if and only if

$$\mathrm{Rel}_{\sqsubseteq^{op}}(F)((f\times g)^{-1}(R)) = (Ff\times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq^{op}}(F)(R)),$$

and therefore F is stable for $\sqsubseteq$ iff it is stable for $\sqsubseteq^{op}$. □

Corollary 2. *The order $\sqsubseteq^{op}$ defining anti-simulations for transition systems as coalgebraic simulations is stable.*

One could conclude from the observation above that there is indeed a natural argument supporting plain similarity as a "right" coalgebraic similarity, definable by a right-stable order. This criterion could be adopted to define right coalgebraic simulations, which plain similarity would satisfy while the opposite relation "is simulated by" would not. However, we immediately noticed that we could define "left-stable" orders by interchanging the roles of Ff and id in the definition of right-stable order, obtaining the inverse inclusion in (1).

Definition 2. *We will say that a functor F with order $\sqsubseteq$ is **left-stable** if, for every function $f : X \longrightarrow Y$, we have*

$$(Ff\times id)^{-1}\sqsubseteq_Y \;\subseteq \coprod_{id\times Ff} \sqsubseteq_X . \tag{5}$$

It is inmediate to check that an order $\sqsubseteq$ is left-stable iff the inverse order $\sqsubseteq^{op}$ is right-stable. Moreover, left-stable orders have the same structural properties that right-stable ones so that, in particular, they are also stable and hence composition-preserving. This is not surprising at all, since the coalgebraic simulations defined by $\sqsubseteq^{op}$ are indeed the opposite of those defined by $\sqsubseteq$ and thus they should have the same structural properties.

But in this case it would be the inverse simulations, corresponding to the "is simulated by" notion, that would be natural instead of plain simulations. As a conclusion, we could use right or left-stability as a criterion to choose a natural direction for the simulation order. But the important fact in both cases is that the simplified asymmetric definitions (using either $\sqsubseteq_X$ or $\sqsubseteq_Y$) of coalgebraic simulations are much easier to handle than the symmetric original definition (where both $\sqsubseteq_X$ and $\sqsubseteq_Y$ have to be used).

4 Covariant-contravariant simulations and conformance simulations

Covariant-contravariant simulations are defined by combining the conditions "to simulate" and "be simulated by", using a partition of the alphabet of actions of the compared labeled transition systems.

Definition 3. *Given $c : X \longrightarrow \mathcal{P}(X)^{Act}$ and $d : Y \longrightarrow \mathcal{P}(Y)^{Act}$ labeled transition systems for the alphabet Act, and $\{Act^r, Act^l, Act^{bi}\}$ a partition of this alphabet, a (Act^r, Act^l)-**simulation** between c and d is a relation $S \subseteq X \times Y$ such that for every $(x, y) \in S$ we have:*

- *for all $a \in Act^r \cup Act^{bi}$ and all $x \xrightarrow{a} x'$ there exists $y \xrightarrow{a} y'$ with $(x', y') \in S$.*
- *for all $a \in Act^l \cup Act^{bi}$, and all $y \xrightarrow{a} y'$ there exists $x \xrightarrow{a} x'$ with $(x', y') \in S$.*

We write $x \; {}_{Act^r}\mathcal{S}_{Act^l} \; y$, and say that x is (Act^r, Act^l)-simulated by y, if and only if there exists some (Act^r, Act^l)-simulation S with xSy.

A very interesting application of this kind of simulations is related with the definition of adequate simulation notions for input/output (I/O) automata [10]. The classic approach to simulations is based on the definition of semantics for reactive systems, where all the actions of the processes correspond to input actions that the user must trigger. Instead, whenever we have explicit output actions the situation is the opposite: it is the system that produces the actions and the user who is forced to accept the produced output. Then, it is natural to conclude that in the simulation framework we have to dualize the simulation condition when considering output actions, and this is exactly what our anti-simulation relations do.

Covariant-contravariant simulations can be easily obtained as coalgebraic simulations, as the following proposition proves.

Proposition 1. *(Act^r, Act^l)-simulations can be defined as the coalgebraic simulations for the functor $F = \mathcal{P}^{Act}$ with functorial order ${}_{Act^r}\sqsubseteq_{Act^l}$ where, for each set X and $\alpha, \alpha' : Act \longrightarrow \mathcal{P}(X)$, we have $\alpha \; {}_{Act^r}\sqsubseteq_{Act^l} \; \alpha'$ if:*

- *for all $a \in Act^r \cup Act^{bi}$, $\alpha(a) \subseteq \alpha'(a)$, and*
- *for all $a \in Act^l \cup Act^{bi}$, $\alpha(a) \supseteq \alpha'(a)$.*

Note that in particular we have $\alpha(a) = \alpha'(a)$ for all $a \in Act^{bi}$.

Proof. Intuitively, using the order ${}_{Act^r}\sqsubseteq_{Act^l}$ on the left-hand side of $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ allows us to remove a'-transitions when $a' \in Act^l$, whereas using it on the right-hand side of $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ allows us to remove a-transitions when $a \in Act^r$.

Let us suppose that we have a classic covariant-contravariant simulation ${}_{Act^r}\mathcal{S}_{Act^l}$ between labeled transition systems $c : P \longrightarrow \mathcal{P}(P)^{Act}$ and $d : Q \longrightarrow \mathcal{P}(Q)^{Act}$ defined by $c(p)(a) = \{p' \mid p \xrightarrow{a} p'\}$ and $d(q)(a) = \{q' \mid q \xrightarrow{a} q'\}$. We must show that if $p \; {}_{Act^r}\mathcal{S}_{Act^l} \; q$ then there exist p^* and q^* such that

$$c(p) \; {}_{Act^r}\sqsubseteq_{Act^l} \; p^* \mathrm{Rel}(\mathcal{P}^{Act})({}_{Act^r}\mathcal{S}_{Act^l}) q^* \; {}_{Act^r}\sqsubseteq_{Act^l} \; d(q). \tag{6}$$

We define p^* and q^* as follows:

– p^* has the same transitions as $c(p)$, except for those transitions $p \xrightarrow{a'} p'$ with $a' \in Act^l$ such that there is no q' with $q \xrightarrow{a'} q'$ and $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. That is, $p^*(a) = \{p' \in c(p)(a) \mid \text{If } a \in Act^l \text{ then exists some } q' \in d(q)(a).\ p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'\}$
– q^* has the same transitions as $d(q)$, except for those transitions $q \xrightarrow{a} q'$ with $a \in Act^r$ such that there is no p' with $p \xrightarrow{a} p'$ and $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Thus, $q^*(a) = \{q' \in d(q)(a) \mid \text{If } a \in Act^r \text{ then exists some } p' \in c(p)(a).\ p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'\}$

It is immediate from these definitions that $c(p) \; {}_{Act^r}\sqsubseteq_{Act^l} \; p^*$ and $q^* \; {}_{Act^r}\sqsubseteq_{Act^l} \; d(q)$, so we are left with checking that $p^*\mathrm{Rel}(\mathcal{P}^{Act})q^*$.

Let $p' \in p^*(a)$ with $a \in Act^r$. By construction of p^*, since we have not dropped any a-transitions from p^*, $p \xrightarrow{a} p'$. Using the fact that ${}_{Act^r}\mathcal{S}_{Act^l}$ is a classic covariant-contravariant simulation, there exists q' such that $q \xrightarrow{a} q'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$, and, again by construction, $q' \in q^*(a)$ because there is some $p \xrightarrow{a} p'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Similarly, if $p' \in p^*(a)$ with $a' \in Act^l$, by construction of p^* there must exist some q' such that $q \xrightarrow{a'} q'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Again, since we have not removed any a'-transitions from $d(q)$ in q^*, it must be true that $q' \in q^*(a)$. Finally, if $p' \in p^*(a)$ with $a \in Act^{bi}$ we have that $p \xrightarrow{a} p'$ and hence there exists q' such that $q \xrightarrow{a} q'$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$, but also $q' \in q^*(a)$.

The argument that shows that for every $q' \in q^*(a)$ there exists some $p' \in p^*(a)$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$ is analogous.

We show now the other implication, that a coalgebraic covariant-contravariant simulation is a classic one. In this case we start from coalgebras c and d that satisfy relation (6) whenever $p \; {}_{Act^r}\mathcal{S}_{Act^l} \; q$.

If $p \xrightarrow{a} p'$ for $a \in Act^r$, then $p' \in p^*(a)$ because $c(p) \; {}_{Act^r}\sqsubseteq_{Act^l} \; p^*$ and, since $p^*\mathrm{Rel}(\mathcal{P}^{Act})({}_{Act^r}\mathcal{S}_{Act^l})q^*$, there is some $q' \in q^*(a)$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$. Again, the definition of ${}_{Act^r}\sqsubseteq_{Act^l}$ ensures that $q^*(a) \subseteq d(q)(a)$ and hence $q \xrightarrow{a} q'$ as required. Similarly, if $q \xrightarrow{a'} q'$ for $a' \in Act^l$, then $q' \in q^*(a)$ because $q^* \; {}_{Act^r}\sqsubseteq_{Act^l} \; d(q)$ and thus, as in the previous case, there exists $p' \in p^*(a)$ with $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$ and $p \xrightarrow{a'} p'$. Finally if $p \xrightarrow{a} p'$ for $a \in Act^{bi}$ (resp. $q \xrightarrow{a} q'$), again by the definition of ${}_{Act^r}\sqsubseteq_{Act^l}$ we have $p' \in p^*(a)$ (resp. $q' \in q^*(a)$) and, from $p^*\mathrm{Rel}(\mathcal{P}^{Act})({}_{Act^r}\mathcal{S}_{Act^l})q^*$, it follows that there exists $q' \in q^*(a)$ (resp. $p' \in p^*(a)$) such that $p' \; {}_{Act^r}\mathcal{S}_{Act^l} \; q'$; by the definition of ${}_{Act^r}\sqsubseteq_{Act^l}$, $q \xrightarrow{a} q'$ (resp. $p \xrightarrow{a} p'$). □

The other new kind of simulations in which we are interested is that of conformance simulations, where the conformance relation in [9,12] meets the simulation world in a nice way. In the definition below we will write $p \xrightarrow{a}$ if $p \xrightarrow{a} p'$ for some p'.

Definition 4. *Given $c : X \longrightarrow \mathcal{P}(X)^A$ and $d : Y \longrightarrow \mathcal{P}(Y)^A$ two labeled transition systems for the alphabet A, a **conformance simulation** between them is a relation $R \subseteq X \times Y$ such that whenever pRq, then:*

– *For all $a \in A$, if $p \xrightarrow{a}$ we must also have $q \xrightarrow{a}$ (this means, using the usual notation for process algebras, that $I(p) \subseteq I(q)$).*

– *For all* $a \in A$ *such that* $q \xrightarrow{a} q'$ *and* $p \xrightarrow{a}$*, there exists some* p' *with* $p \xrightarrow{a} p'$ *and* $p' R q'$*.*

Conformance simulations allow the extension of the set of actions offered by a process, so that in particular we will have $a < a + b$, but they also consider that a process can be "improved" by reducing the nondeterminism in it, so that $ap + aq < ap$. In this way we have again a kind of covariant-contravariant simulation, not driven by the alphabet of actions executed by the processes but by their nondeterminism.

Once again, conformance simulations can be defined as coalgebraic simulations taking the adequate order on the functor defining labeled transition systems.

Proposition 2. *Conformance simulations can be obtained as the coalgebraic simulations for the order* $\sqsubseteq^{Conf}$ *on the functor* $\mathcal{P}^A$*, where for any set* X *we have* $u \sqsubseteq_X^{Conf} v$ *if for every* $u, v : A \longrightarrow \mathcal{P}X$ *and* $a \in A$*:*

– *either* $u(a) = \emptyset$*, or*
– $u(a) \supseteq v(a)$ *and* $v(a) \neq \emptyset$*.*

Proof. The proof is similar to that of Proposition 1. Let us first prove that $\sqsubseteq_X^{Conf}$ is indeed an order. It is clear that the only not immediate property is transitivity. To check it, let us take $u \sqsubseteq_X^{Conf} v \sqsubseteq_Y^{Conf} w$: if $u(a) = \emptyset$ we are done; otherwise, we have $u(a) \supseteq v(a)$ and $v(a) \neq \emptyset$, so that we also have $v(a) \supseteq w(a)$ and $w(a) \neq \emptyset$, obtaining $u(a) \supseteq w(a)$ and $w(a) \neq \emptyset$.

Now, we can interpret that using the order $\sqsubseteq^{Conf}$ on the left-hand side of $\mathrm{Rel}_{\sqsubseteq}(F)(R)$ allows us to remove all a-transitions except for the last one, whereas using it on the right-hand side allows us to remove all b-transitions for $b \in B$, where B is any set of actions. First, let us suppose that we have a classic conformance simulation R^C between the transition systems $\mathcal{A} = (P, A, \rightarrow_{\mathcal{A}})$ and $\mathcal{A}' = (Q, A, \rightarrow_{\mathcal{A}'})$, with pR^Cq. We will show that R^C is also a coalgebraic conformance simulation.

As usual, take $c : P \longrightarrow \mathcal{P}(P)^A$ and $d : Q \longrightarrow \mathcal{P}(Q)^A$ defined by $c(p)(a) = \{p' \mid p \xrightarrow{a} p'\}$ and $d(q)(a) = \{q' \mid q \xrightarrow{a} q'\}$. We must show that if pR^Cq then there exist p^* and q^* such that

$$c(p) \sqsubseteq^{Conf} p^* \mathrm{Rel}(\mathcal{P}(id)^A)(R^C) q^* \sqsubseteq^{Conf} d(q). \tag{7}$$

We define p^* and q^* in the following way:

– p^* has the same transitions as $c(p)$ except for those $p \xrightarrow{a} p'$ such that there is no q' with $q \xrightarrow{a'} q'$ and $p' R^C q'$.
– $q^*(a)$ is the same as $d(q)(a)$ except when $d(q)(a) \neq \emptyset$ and $c(p)(a) = \emptyset$, in which case $q^*(a) = \emptyset$.

Since either $q^*(a) = d(q)(a)$ or $q^*(a) = \emptyset$, it is obvious that $q^* \sqsubseteq^{Conf} d(q)$. To check that $c(p) \sqsubseteq^{Conf} p^*$ we must ensure, for each $a \in A$, that we do not remove every a-transition of p^*, that is, $p^*(a) \neq \emptyset$. But since R^C is a classic

conformance simulation, if $p \xrightarrow{a}$ then $q \xrightarrow{a} q'$ for some q' and, moreover, there must exist an a-successor p' such that $p' R^C q'$. By construction of p^*, $p' \in p^*(a)$.

Let us see now that $p^* \mathrm{Rel}(\mathcal{P}^A)(R^C) q^*$, that is, $p^*(a) \mathrm{Rel}(\mathcal{P}) q^*(a)$ for every $a \in A$. First, assume that $p' \in p^*(a)$; then, by construction of p^*, there is some q' such that $q \xrightarrow{a} q'$ and $p' R^C q'$. But, since $p \xrightarrow{a}$, we have by construction that $q^*(a) = d(q)(a) \ni q'$, as we needed to prove. Analogously, assume $q' \in q^*(a)$. Since $q^*(a) \neq \emptyset$, by construction we have $q^*(a) = d(q)(a)$ and $c(p)(a) \neq \emptyset$, and therefore $q \xrightarrow{a} q'$ and also $p \xrightarrow{a}$. Now, using the fact that R^C is a classic conformance simulation, there exists p' such that $p \xrightarrow{a} p'$ with $p' R^C q'$ and thus $p' \in p^*(a)$, as required.

We show now the other implication, that a coalgebraic conformance simulation is a classic one. In this case we start from coalgebras c and d that satisfy relation (7) whenever $p R^C q$.

First, we will show that if $p \xrightarrow{a}$ then $q \xrightarrow{a}$ as well. Indeed, $p \xrightarrow{a}$ means that $c(p)(a) \neq \emptyset$ and, since $c(p) \sqsubseteq^{Conf} p^*$, it follows that $p^*(a) \neq \emptyset$. By definition of $\mathrm{Rel}(\mathcal{P}^A)$ this implies that $q^*(a) \neq \emptyset$ and, since $q^* \sqsubseteq^{Conf} d(q)$, $d(q)(a) \neq \emptyset$, so $q \xrightarrow{a}$ as we wanted to show.

In the second place, let us assume that $q \xrightarrow{a} q'$ and that $p \xrightarrow{a}$. Since $q^* \sqsubseteq^{Conf} d(q)$, either $q^*(a) = \emptyset$ or $q^*(a) \supseteq d(q)(a) \ni q'$; arguing as above, $p \xrightarrow{a}$ leads to $q^*(a) \neq \emptyset$, so the latter must hold. Now, from $p^* \mathrm{Rel}(\mathcal{P}^{\mathcal{A}}) q^*$ it follows that there exists $p' \in p^*(a)$ with $p' R^C q'$ and, since $c(p) \sqsubseteq^{Conf} p^*$ and $c(p)(a) \neq \emptyset$, we have $c(p)(a) \supseteq p^*(a) \ni p'$, that is, $p \xrightarrow{a} p'$ as required. □

Next we check that the order ${}_{Act^r}\sqsubseteq_{Act^l}$ defining covariant-contravariant simulations is stable.

Lemma 3. *Given a partition $\{Act^r, Act^l, Act^{bi}\}$ of Act the order ${}_{Act^r}\sqsubseteq_{Act^l}$ for the functor $\mathcal{P}^{Act}$ defining covariant-contravariant simulations for transition systems is stable.*

Proof. It is clear that the order ${}_{Act^r}\sqsubseteq_{Act^l}$ can be obtained as the product of a family of orders $\sqsubseteq^a$ for the functor $\mathcal{P}$, with $a \in Act$. This is indeed the case taking $\sqsubseteq^a_X = \subseteq_X$ for $a \in Act^r$, $\sqsubseteq^a_X = \supseteq_X$ for $a \in Act^l$ and $\sqsubseteq^a_X = =_X$ for $a \in Act^{bi}$. Then it is easy to see that to obtain that ${}_{Act^r}\sqsubseteq_{Act^l}$ is stable it is enough to prove that each of the orders $\sqsubseteq^a$ is stable.

This latter requirement is straightforward because, for $a \in Act^r$, $\sqsubseteq^a$ is right-stable; for $a \in Act^l$ the order $\sqsubseteq^a$ is left-stable; and for $a \in Act^{bi}$, $\sqsubseteq^a$ is the equality relation, which is both right and left-stable, for every functor F. □

Certainly, the order defining covariant-contravariant simulations is not right-stable nor left-stable, but in the proof above we have used the power of these two properties thanks to the fact that the order ${}_{Act^r}\sqsubseteq_{Act^l}$ can be factorised as the product of a family of orders that are either right-stable or left-stable. Then we can obtain the following sequence of general definitions and results, from which Lemma 3 could be obtained as a simple particular case.[3]

[3] Instead of removing the above, we have preferred to maintain the sequence of results in the order in which we got them, starting with our motivating example.

Definition 5. *We say that an order $\sqsubseteq$ on a functor F^A is* ***action-distributive*** *if there is a family of orders $\sqsubseteq^a$ on F such that*

$$f \sqsubseteq g \iff f(a) \sqsubseteq^a g(a) \text{ for all } a \in A.$$

Whenever $\sqsubseteq$ can be distributed in this way we will write $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$.

Definition 6. *We say that an action-distributive order $\sqsubseteq$ on F^A is* ***side stable*** *if for the decomposition $\sqsubseteq = \prod_{a\in A} \sqsubseteq^a$ we have that each order $\sqsubseteq^a$ is either right-stable or left-stable.*

By separating the right-stable and the left-stable components we obtain $\sqsubseteq = \sqsubseteq^l \times \sqsubseteq^r$, where A^r (resp. A^l) collects the set of arguments[4] $a \in A$ with $\sqsubseteq^a$ right-stable (resp. left-stable). We extend $\sqsubseteq^l$ and $\sqsubseteq^r$ to obtain a pair of orders on F^A, $\sqsubseteq^{\bar{l}}$ and $\sqsubseteq^{\bar{r}}$, defined by:

- *$f \sqsubseteq^{\bar{r}} g$ iff $f(a) \sqsubseteq^a g(a)$ for all $a \in A^r$ and $f(a) = g(a)$ for all $a \in A^l$.*
- *$f \sqsubseteq^{\bar{l}} g$ iff $f(a) \sqsubseteq^a g(a)$ for all $a \in A^l$ and $f(a) = g(a)$ for all $a \in A^r$.*

Proposition 3. *The order $\sqsubseteq^{\bar{l}}$ is left-stable, while $\sqsubseteq^{\bar{r}}$ is right-stable. We have $\sqsubseteq = (\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}}) = (\sqsubseteq^{\bar{r}} \circ \sqsubseteq^{\bar{l}})$, and therefore we also have $\sqsubseteq = (\sqsubseteq^{\bar{l}} \cup \sqsubseteq^{\bar{r}})^*$.*

Proof. In first place it is obvious that $\sqsubseteq^{\bar{r}}$ (resp. $\sqsubseteq^{\bar{l}}$) is right-stable (resp. left-stable) since it is defined either by a right-stable (resp. left-stable) order or the equality relation.

Now, let us show that $\sqsubseteq = (\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}})$. Let us suppose that $f \sqsubseteq g$ then, by definition, $f(a) \sqsubseteq^a g(a)$ for all $a \in A$. Let us show that there is some h such that $f \sqsubseteq^{\bar{r}} h$ and $h \sqsubseteq^{\bar{l}} g$. We define $h(a)$ the following way: if $a \in A^r$ $h(a) = g(a)$; and if $a \in A^l$ $h(a) = f(a)$. Defined like this it is obvious that $f \sqsubseteq^{\bar{r}} h$ and $h \sqsubseteq^{\bar{l}} g$. For example if $a \in A^r$ then since $f \sqsubseteq g$, $f(a) \sqsubseteq^a g(a) = h(a)$ and if $a \in A^l$ $f(a) = h(a)$, that is, $f \sqsubseteq^{\bar{r}} h$.

Now, if we take f $(\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}})$ g there is some $h \in F^A$ such that $f \sqsubseteq^{\bar{r}} h$ and $h \sqsubseteq^{\bar{l}} g$. That is, on one hand if $a \in A^r$ then, by definition, $f(a) \sqsubseteq^a h(a)$ and $h(a) = g(a)$, on the other hand if $a \in A^l$ $f(a) = h(a)$ and $h(a) \sqsubseteq^a g(a)$. So, we have that $f \sqsubseteq g$.

The prove for $(\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}}) = (\sqsubseteq^{\bar{r}} \circ \sqsubseteq^{\bar{l}})$ is similar. □

Proposition 4. *For any side stable order $\sqsubseteq$ on F^A, if we have a decomposition $\sqsubseteq = \sqsubseteq^l \times \sqsubseteq^r$ based on a partition of A into a set of right-stable components A^r and another set of left-stable components A^l, then we can obtain the coalgebraic simulations for $\sqsubseteq$ as the $(\sqsubseteq^{\bar{r}}_Y \circ \mathrm{Rel}(F) \circ \sqsubseteq^{\bar{l}}_X)$-coalgebras.*

[4] We have assumed here a partition $\{A^l, A^r\}$ of the set A into two sets of right-stable and left-stable components. Obviously, if there were some arguments $a \in A$ on which $\sqsubseteq^a$ is both right-stable and left-stable then the decomposition would not be unique, but the result would be valid for any such decomposition.

Proof. By definition, $\mathrm{Rel}_{\sqsubseteq}(F)(R) = \sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X$. Also, since $\sqsubseteq = (\sqsubseteq^{\bar{r}} \circ \sqsubseteq^{\bar{l}}) = (\sqsubseteq^{\bar{l}} \circ \sqsubseteq^{\bar{r}})$, we have:

$$\begin{aligned}
\sqsubseteq_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq_X &= (\sqsubseteq^{\bar{l}}_Y \circ \sqsubseteq^{\bar{r}}_Y) \circ \mathrm{Rel}(F)(R) \circ (\sqsubseteq^{\bar{r}}_X \circ \sqsubseteq^{\bar{l}}_X) \\
&= \sqsubseteq^{\bar{l}}_Y \circ (\sqsubseteq^{\bar{r}}_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{r}}_X) \circ \sqsubseteq^{\bar{l}}_X \\
&= (\sqsubseteq^{\bar{l}}_Y \circ \sqsubseteq^{\bar{r}}_Y) \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{l}}_X \\
&\qquad \text{(by right-stability of } \sqsubseteq^{\bar{r}}) \\
&= \sqsubseteq^{\bar{r}}_Y \circ (\sqsubseteq^{\bar{l}}_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{l}}_X) \\
&\qquad \text{(since } \sqsubseteq^{\bar{r}} \text{ and } \sqsubseteq^{\bar{l}} \text{ commute)} \\
&= \sqsubseteq^{\bar{r}}_Y \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^{\bar{l}}_X \\
&\qquad \text{(by left-stability of } \sqsubseteq^{\bar{l}})
\end{aligned}$$

□

The characterization above still requires the use of the order on both sides of the $\mathrm{Rel}(F)(R)$ operator. However, the fact that $\sqsubseteq^{\bar{r}}_Y$ (resp. $\sqsubseteq^{\bar{l}}_X$) is right-stable (resp. left-stable) makes the application of this decomposition as simple as when coping with either a right or left-stable order.

Proposition 5. *If $\sqsubseteq = \prod_{a \in A} \sqsubseteq^a$ and $\sqsubseteq^a$ is stable for all $a \in A$, then $\sqsubseteq$ is stable.*

Proof. The result follows from the following chain of implications:

$$\begin{aligned}
& (u, v) \in (Ff \times Fg)^{-1} \mathrm{Rel}_{\sqsubseteq}(F)(R) \\
\iff\ & Ff(u) \sqsubseteq z' \mathrm{Rel}(F)(R) w' \sqsubseteq Fg(v) \\
\iff\ & Ff(u)(a) \sqsubseteq^a z'(a) \mathrm{Rel}(F^a)(R) w'(a) \sqsubseteq^a Fg(v)(a), \text{ for all } a \\
\iff\ & (u(a), v(a)) \in (Ff \times Fg)^{-1} \mathrm{Rel}_{\sqsubseteq^a}(F)(R), \text{ for all } a \\
\implies\ & (u(a), v(a)) \in \mathrm{Rel}_{\sqsubseteq^a}(F)((f \times g)^{-1} R), \text{ for all } a \\
\iff\ & u(a) \sqsubseteq^a x'(a) \mathrm{Rel}(F)((f \times g)^{-1} R) y'(a) \sqsubseteq^a v(a), \text{ for all } a \\
\iff\ & (u, v) \in \mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1} R)
\end{aligned}$$

□

Corollary 3. *Any side stable order is stable.*

Corollary 4. *The order ${}_{Act^r}\sqsubseteq_{Act^l}$ defining covariant-contravariant simulations is side stable and therefore it is stable too.*

Next we consider the case of conformance simulations, for which we can obtain similar results to those proved for covariant-contravariant simulations.

Lemma 4. *The order $\sqsubseteq^{Conf}$ defining conformance simulations for transition systems is stable.*

Proof. Let $R \subseteq Z \times W$ be a relation and $f : X \longrightarrow Z$, $g : Y \longrightarrow W$ arbitrary functions. If $(u, v) \in (\mathcal{P}^A f \times \mathcal{P}^A g)^{-1}(\mathrm{Rel}_{\sqsubseteq^{Conf}}(\mathcal{P}^A)(R))$, then there exist z and w such that

$$\mathcal{P}^A f(u) \sqsubseteq^{Conf} z \ \mathrm{Rel}(\mathcal{P}^A)(R) \ w \sqsubseteq^{Conf} \mathcal{P}^A g(v). \tag{8}$$

We have to show that $(u, v) \in \mathrm{Rel}_{\sqsubseteq^{Conf}}(\mathcal{P}^A)((f \times g)^{-1}(R))$, that is, there exist x and y such that

$$u \sqsubseteq^{Conf} x \ \mathrm{Rel}(\mathcal{P}^A)((f \times g)^{-1}(R)) \ y \sqsubseteq^{Conf} v.$$

Let us define $x : A \longrightarrow \mathcal{P}(X)$ by $x(a) = u(a) \cap f^{-1}(z(a))$ and $y : A \longrightarrow \mathcal{P}(Y)$ by $y(a) = g^{-1}(w(a))$. Then we have:

1. $u \sqsubseteq^{Conf} x$.
   If $u(a) = \emptyset$, there is nothing to prove. Otherwise, since $\mathcal{P}^A f(u) \sqsubseteq^{Conf} z$ and $f(u(a)) \neq \emptyset$, we have $f(u(a)) \supseteq z(a) \neq \emptyset$ and hence $u(a) \supseteq u(a) \cap f^{-1}(z(a)) = x(a) \neq \emptyset$.
2. $y \sqsubseteq^{Conf} v$.
   If $w(a) = \emptyset$, then $y(a) = g^{-1}(w(a)) = \emptyset$. Otherwise, since $w \sqsubseteq^{Conf} \mathcal{P}^A g(v)$, we have $w(a) \supseteq g(v(a)) \neq \emptyset$, so that $v(a) \neq \emptyset$ and $y(a) = g^{-1}(w(a)) \supseteq g^{-1}(g(v(a))) \supseteq v(a)$.
3. $x \ \mathrm{Rel}(\mathcal{P}^A)((f \times g)^{-1}(R)) \ y$.
   For every $a \in A$ we need to show that $x(a) \ \mathrm{Rel}(\mathcal{P})((f \times g)^{-1}(R)) \ y(a)$, which means:
   (a) for every $p \in x(a)$ there exists $q \in y(a)$ such that $p \ (f \times g)^{-1}(R) \ q$, that is, $f(p) R g(q)$; and
   (b) for every $q \in y(a)$ there exists $p \in x(a)$ such that $p \ (f \times g)^{-1}(R) \ q$, that is, $f(p) R g(q)$.
   In the first case, let $p \in x(a)$; by definition of x, $f(p) \in z(a)$. Now, from $z \ \mathrm{Rel}(\mathcal{P}^A)(R) \ w$ we obtain that for each $p' \in z(a)$ there exists $q' \in w(a)$ such that $p' R q'$. Then, for $f(p) \in z(a)$ there exists $q' \in w(a)$ with $f(p) R q'$; and by definition of y, there exists $q \in y(a)$ with $q' = g(q)$ as required.
   In the second case, let $q \in y(a)$ so that $g(q) \in w(a)$. Again, from $z \ \mathrm{Rel}(\mathcal{P}^A)(R) \ w$ it follows that there is $p' \in z(a)$ with $p' R g(q)$. Now, $f(u(a)) \supseteq z(a)$ because $u \sqsubseteq^{Conf} z$, so there exists $p \in u(a) \cap f^{-1}(z(a))$ with $f(p) = p'$, as required. □

As in the case of covariant-contravariant simulations, conformance simulations cannot be defined as coalgebraic simulations using neither a right-stable order nor a left-stable order. But we can find in the arguments above the basis for a decomposition of the involved order $\sqsubseteq^{Conf}$, according to the two cases in its definition. Once again $\sqsubseteq^{Conf}$ is an action-distributive order on $\mathcal{P}^A$, but in order to obtain the adequate decomposition of $\sqsubseteq^{Conf}$ now we also need to decompose the component orders $\sqsubseteq^a$.

Definition 7. *We define the conformance orders $\sqsubseteq^{C\neg\emptyset}$, $\sqsubseteq^{C\emptyset}$, and $\sqsubseteq^{C}$ on the functor $\mathcal{P}$ by:*

- $x_1 \sqsubseteq^{C\emptyset} x_2$ *if* $x_1 = \emptyset$ *or* $x_1 = x_2$.
- $x_1 \sqsubseteq^{C\neg\emptyset} x_2$ *if* $x_1 \supseteq x_2$ *and* $x_2 \neq \emptyset$, *or* $x_1 = x_2$.
- $x_1 \sqsubseteq^{C} x_2$ *if* $x_1 \sqsubseteq^{C\neg\emptyset} x_2$ *or* $x_1 \sqsubseteq^{C\emptyset} x_2$.

Proposition 6. *The two relations $\sqsubseteq^{C\emptyset}$ and $\sqsubseteq^{C\neg\emptyset}$ commute with each other:*

$$(\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset}) = (\sqsubseteq^{C\neg\emptyset} \circ \sqsubseteq^{C\emptyset}),$$

from where it follows that $(\sqsubseteq^{C\emptyset} \cup \sqsubseteq^{C\neg\emptyset})^* = (\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset}) = (\sqsubseteq^{C\neg\emptyset} \circ \sqsubseteq^{C\emptyset})$. We also have $\sqsubseteq^{C} = (\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset})$, from where we conclude that $\sqsubseteq^{C}$ is indeed an order relation.

Proof. Let $u\ (\sqsubseteq^{C\emptyset} \circ \sqsubseteq^{C\neg\emptyset})\ v$: there is some w such that $u \sqsubseteq^{C\neg\emptyset} w$ and $w \sqsubseteq^{C\emptyset} v$. We need to find w' such that $u \sqsubseteq^{C\emptyset} w'$ and $w' \sqsubseteq^{C\neg\emptyset} v$. If $w = \emptyset$ then it must be $u = \emptyset$ too, and we can take $w' = v$; otherwise, it must be $v = w$ and we can take $w' = u$. The other inclusion is similar. □

Proposition 7. *The order $\sqsubseteq^{C\neg\emptyset}$ is right-stable whereas the order $\sqsubseteq^{C\emptyset}$ is left-stable.*

Proof. Let us see in first place that $\sqsubseteq^{C\emptyset}$ is right-stable. Let $(v, u) \in (id \times Ff)^{-1}\ \sqsubseteq^{C\emptyset}$, that is, by definition of $\sqsubseteq^{C\emptyset}$ we have that $v = \emptyset$ or $v = Ff(u)$. We must prove that $(v, u) \in \coprod_{(Ff\times id)} \sqsubseteq^{C\emptyset}$, that is, that there is some u' such that $v = Ff(u')$ and $u' \sqsubseteq^{C\emptyset} u$. In first place since $(v, u) \in (id \times Ff)^{-1}\ \sqsubseteq^{C\emptyset}$ then $v = \emptyset$ or $v = Ff(u)$ so, in the first case $u' = \emptyset$ and in the second we can take $u' = u$.

Now we are going to see that $\sqsubseteq^{C\neg\emptyset}$ is left-stable, so let us suppose that $(u, v) \in (Ff \times id)^{-1}\ \sqsubseteq^{C\neg\emptyset}$. In this case either $Ff(u) = v$ or $Ff(u) \supseteq v$ with $v \neq \emptyset$. We have to find some v' such that $u \sqsubseteq^{C\neg\emptyset} v'$ with $Ff(v') = v$. If $Ff(u) = v$ it is trivially true for $v' = u$; but in the other case since $v \subseteq Ff(u)$ there must be some $v' \subseteq u$ such that $Ff(v') = v$. □

Corollary 5. *The order $\sqsubseteq^{Conf}$ defining conformance simulations can be decomposed into $\prod_{a\in A} \sqsubseteq^{a}$ where, for each $a \in A$, we have $\sqsubseteq^{a} = \sqsubseteq^{C}$ as defined above. Then, $\sqsubseteq^{Conf} = \prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset} \cup \sqsubseteq^{a,\emptyset})^* = \prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset}) \circ \prod_{a\in A}(\sqsubseteq^{a,\emptyset}) = \prod_{a\in A}(\sqsubseteq^{a,\emptyset}) \circ \prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset})$, so that we obtain $\sqsubseteq^{Conf}$ as the composition of a right-stable order and a left-stable order that commute with each other.*

Proposition 8. *For any pair of right (resp. left)-stable orders $\sqsubseteq^1$, $\sqsubseteq^2$ on F, their composition also defines a right (resp. left)-stable order on F.*

Proof. Given $f : X \longrightarrow Y$ we must show that

$$(id \times Ff)^{-1}(\sqsubseteq^1_Y \circ \sqsubseteq^2_Y) \subseteq \coprod_{(Ff\times id)} (\sqsubseteq^1_X \circ \sqsubseteq^2_X).$$

Let us assume that $(y, x) \in (id \times Ff)^{-1}(\sqsubseteq^1_Y \circ \sqsubseteq^2_Y)$, that is, $y\ (\sqsubseteq^1 \circ \sqsubseteq^2)\ y' = Ff(x)$; then, there exists $y'' \in FY$ such that $y \sqsubseteq^2_Y y''$ and $y'' \sqsubseteq^1_Y y'$. Graphically,

$$\begin{array}{ccccc} y & \sqsubseteq^2_Y & y'' & \sqsubseteq^1_Y & y' \\ & & & & \uparrow Ff \\ & & & & x \end{array} \tag{9}$$

Since $\sqsubseteq^1_Y$ is right-stable we have that $(id \times Ff)^{-1}\ \sqsubseteq^1_Y \subseteq \coprod_{(Ff \times id)} \sqsubseteq^1_X$. Hence, there exists $x'' \in FX$ such that $Ff(x'') = y''$ and $x''\ \sqsubseteq^1_X\ x$, thus turning diagram (9) into the following:

$$\begin{array}{ccc} y & \sqsubseteq^2_Y & y'' \\ & & \uparrow{\scriptstyle Ff} \\ & & x'' \quad \sqsubseteq^1_X \quad x \end{array} \tag{10}$$

Now, we can apply right-stability of $\sqsubseteq^2$: since we have $(y, x'') \in (id \times Ff)^{-1}\ \sqsubseteq^2_Y \subseteq \coprod_{(Ff \times id)} \sqsubseteq^2_X$, there exists $x' \in FX$ such that $Ff(x') = y$ and $x'\ \sqsubseteq^2_X\ x''$. Thus, diagram (10) becomes

$$\begin{array}{ccccc} y & & & & \\ \uparrow{\scriptstyle Ff} & & & & \\ x' & \sqsubseteq^2_X & x'' & \sqsubseteq^1_X & x \end{array} \tag{11}$$

which means that there exist $x', x'' \in FX$ such that $Ff(x') = y$, $x'\ \sqsubseteq^2_X\ x''$ and $x''\ \sqsubseteq^1_X\ x$, or equivalently, that $(y, x) \in \coprod_{(Ff \times id)}(\sqsubseteq^1_X \circ \sqsubseteq^2_X)$, as we had to prove. □

Proposition 9. *If $\sqsubseteq^r$ is a right-stable order on F and $\sqsubseteq^l$ is a left-stable order on F that commute with each other, then their composition defines a stable order on F. Moreover, the coalgebraic simulations for the order $\sqsubseteq = \sqsubseteq^r \circ \sqsubseteq^l$ can be equivalently defined as the $(\sqsubseteq^r \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq^l)$-coalgebras.*

Proof. Let $R \subseteq Z \times W$ be a relation, $f : X \longrightarrow Z$ and $g : Y \longrightarrow W$ arbitrary functions, and $\sqsubseteq\ = \sqsubseteq^r \circ \sqsubseteq^l$. Let us suppose that $(u, v) \in (Ff \times Fg)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$. Then, since $\sqsubseteq^r$ and $\sqsubseteq^l$ commute with each other, using Proposition 4, there exist z', w' such that

$$Ff(u)\ \sqsubseteq^l_Z\ z'\ \mathrm{Rel}(F)(R)\ w'\ \sqsubseteq^r_W\ Fg(v). \tag{12}$$

If we write z for $Ff(u)$ and w for $Fg(v)$, then equation (12) is equivalent to

$$\begin{array}{ccccccccc} z & \sqsubseteq^l_Z & z' & \mathrm{Rel}(F)(R) & w' & \sqsubseteq^r_W & w \\ {\scriptstyle Ff}\uparrow & & & & & & \uparrow{\scriptstyle Fg} \\ u & & & & & & v \end{array} \tag{13}$$

and we have to show that $(u, v) \in \mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R))$, that is, that there exist x and y such that

$$u\ \sqsubseteq^l_X\ x\ \mathrm{Rel}(F)((f \times g)^{-1}(R))\ y\ \sqsubseteq^r_Y\ v.$$

Using that $\sqsubseteq^r$ is right-stable on the rhs of equation (12), we get $(w', v) \in (id \times Fg)^{-1}\ \sqsubseteq^r_W\ \subseteq\ \coprod_{(Fg \times id)} \sqsubseteq^r_Y$, so that there is some $y \in FY$ such that

$Fg(y) = w'$, with $y \sqsubseteq^r_Y v$. Graphically, diagram (13) becomes

$$\begin{array}{ccccccc} z & \sqsubseteq^l_Z & z' & \mathrm{Rel}(F)(R) & w' & & \\ {\scriptstyle Ff}\uparrow & & & & \uparrow{\scriptstyle Fg} & & \\ u & & & & y & \sqsubseteq^r_Y & v \end{array} \tag{14}$$

Analogously, applying the left-stability of order $\sqsubseteq^l_Z$ we get that there is some $x \in FX$ with $Ff(x) = z'$ such that $u \sqsubseteq^l_X x$. Or graphically,

$$\begin{array}{ccccccc} & & z' & \mathrm{Rel}(F)(R) & w' & & \\ & & {\scriptstyle Ff}\uparrow & & \uparrow{\scriptstyle Fg} & & \\ u & \sqsubseteq^l_X & x & & y & \sqsubseteq^r_Y & v \end{array} \tag{15}$$

But diagram (15) is just what we had to prove, since we have found x, y such that $(x, y) \in (Ff \times Fg)^{-1}(\mathrm{Rel}(F)(R)) = \mathrm{Rel}(F)((f \times g)^{-1}(R))$ with $u \sqsubseteq^l_X x$, $y \sqsubseteq^r_Y v$ or, in other words, $(u, v) \in \mathrm{Rel}_{\sqsubseteq}(F)((f \times g)^{-1}(R))$. □

In particular, for our running example of conformance simulations we obtain the corresponding factorization of the definition of coalgebraic simulations for the order $\sqsubseteq^{Conf}$:

Corollary 6. *Coalgebraic simulations for the conformance order $\sqsubseteq^{Conf}$ can be equivalently defined as the $(\prod_{a\in A}(\sqsubseteq^{a,\neg\emptyset}_Y) \circ \mathrm{Rel}(F)(R) \circ \prod_{a\in A}(\sqsubseteq^{a,\emptyset}_X))$-coalgebras.*

5 Conclusion

We have presented in this paper two new simulation orders induced by two criteria that capture the difference between input and output actions and the implementation notions that are formalized by the conformance relations.

In order to apply the general theory of coalgebraic simulations to them, we identified the corresponding orders on the functor defining labeled transition systems. However, it was not immediate to prove that the obtained orders had the desired good properties since the usual way to do it, namely, by establishing stability as a consequence of a stronger property that we have called right-stability, is not applicable in this case.

Trying to adapt that property to our situation we have discovered several interesting consequences. We highlight the fact that right-stability is an assymetric property which has proved to be very useful for the study of a "reversible" concept such as that of relation, since it is clear that any structural result on the theory of relations should remain true when we reverse the relations, simply "observing" them "from the other side". Two consequences of that assymetric approach followed: first we noticed that we could use it to point the simulation orders in some natural way; secondly we also noticed that by dualizing the right-stability condition we could obtain left-stability.

But the crucial result in order to be able to manage more complicated simulation notions, as proved to be the case for our new covariant-contravariant simulations and the conformance simulations, was the discovery of the fact that both of them could be factorized into the composition of a right-stable and a left-stable component. Exploiting this decomposition we have been able to easily adapt all the techniques that had proved to be very useful for the case of right-stable orders.

We plan to expand our work here in two different directions. The first one is concerned with the two new simulated notions introduced in this paper: once we know that they can be defined as stable coalgebraic simulations and therefore have all the desired basic properties of simulations, we will continue with their study by integrating them into our unified presentation of the semantics for processes [4]. Hence we expect to obtain, in particular, a clear relation between conformance similarity and the classic similarity orders as well as an algebraic characterization for the new semantics. In addition, we plan to continue with our study of stability, which has proved to be a crucial property in order to understand the notion of coalgebraic simulation, thus making it possible to apply the theory to other examples like those studied in this paper.

References

1. P. Aczel and N. P. Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, vol 389 of *LNCS*, pages 357–365. Springer, 1989.
2. B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.
3. D. de Frutos-Escrig and C. Gregorio-Rodríguez. Universal coinductive characterisations of process semantics. In G. Ausiello, J. Karhumäki, G. Mauri, and C.-H. L. Ong, editors, *IFIP TCS*, vol 273 of *IFIP*, pages 397–412. Springer, 2008.
4. D. de Frutos Escrig, C. Gregorio-Rodríguez, and M. Palomino. On the unification of semantics for processes: observational semantics. In M. Nielsen, A. Kucera, P. Bro Miltersen, C. Palamidessi, P. Tuma, and F. Valencia, editors, *SOFSEM 09: Theory and Practice of Computer Science. 35th International Conference on Current Trends in Theory and Practice of Computer Science, Proceedings*, vol 5404 of *LNCS*, pages 279–290. Springer, 2009.
5. J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. Comput.*, 100(2):202–260, 1992.
6. J. Hughes and B. Jacobs. Simulations in coalgebra. *TCS*, 327(1-2):71–108, 2004.
7. B. Jacobs and J. Hughes. Simulations in coalgebra. In H. P. Gumm, editor, *CMCS'03: 6th International Workshop on Coalgebraic Methods in Computer Science*, volume 82, 2003.
8. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.
9. G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. *Computer Networks and ISDN Systems*, 25(1):23–41, 1992.
10. N. A. Lynch and M. R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, 1987.

11. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science, 5th GI-Conference, Proceedings*, vol 104 of *LNCS*, pages 167–183. Springer, 1981.
12. J. Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.
13. R. J. van Glabbeek. The linear time-branching time spectrum I: The semantics of concrete, sequential processes. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of process algebra*, pages 3–99. 2001.

On the specification of modal systems: a comparison of three frameworks[☆]

Luca Aceto[a], Ignacio Fábregas[b], David de Frutos-Escrig[b], Anna Ingólfsdóttir[a], Miguel Palomino[b]

[a] *ICE-TCS, School of Computer Science, Reykjavik University, Iceland*
[b] *Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Spain*

Abstract

This paper studies the relationships between three notions of behavioural preorder that have been proposed in the literature: refinement over modal transition systems, and the covariant-contravariant simulation and the partial bisimulation preorders over labelled transition systems. It is shown that there are mutual translations between modal transition systems and labelled transition systems that preserve, and reflect, refinement and the covariant-contravariant simulation preorder. The translations are also shown to preserve the modal properties that can be expressed in the logics that characterize those preorders. A translation from labelled transition systems modulo the partial bisimulation preorder into the same model modulo the covariant-contravariant simulation preorder is also offered, together with some evidence that the former model is less expressive than the latter. In order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, their connections are also phrased and studied in the context of institutions.

1. Introduction

Modal transition systems (MTSs) have been proposed in, e.g., [16, 17] as a model of reactive computation based on states and transitions that naturally supports a notion of *refinement* that is akin to the notion of implication in logical specification languages. (See the paper [6] for a thorough analysis of the connections between specifications given in terms of MTSs and logical specifications in the setting of a modal logic that characterizes refinement.) In an MTS,

[☆]Research supported by Spanish projects DESAFIOS10 TIN2009-14599-C03-01, TESIS TIN2009-14321-C02-01 and PROMETIDOS S2009/TIC-1465, and the NILS Mobility Project (Abel Extraordinary Chair programme). Luca Aceto and Anna Ingólfsdóttir have been partially supported by the project 'Processes and Modal Logics' (project nr. 100048021) of the Icelandic Fund for Research.

transitions come in two flavours: the *may* transitions and the *must* transitions, with the requirement that each must transition is also a may transition. The idea behind the notion of refinement over MTSs is that, in order to implement correctly a specification, an implementation should exhibit all the transitions that are required by the specification (these are the must transitions in the MTS that describes the specification) and may provide the transitions that are allowed by the specification (these are the may transitions in the MTS that describes the specification).

The formalism of modal transition systems is intuitive, has several variants with varying degrees of expressive power and complexity—see, e.g., the survey paper [3]—and has recently been used as a suitable model for the specification of service-oriented applications. In particular, results on the supervisory control (in the sense of Ramadge and Wonham [20]) of systems whose specification is given in that formalism have been presented in, e.g., [7, 11].

The very recent development of the notion of *partial bisimulation* in the setting of labelled transition systems (LTSs) presented in [4, 5] has been explicitly motivated by the desire to develop a process-algebraic model within which one can study topics in the field of supervisory control. A partial bisimulation is a variation on the classic notion of bisimulation [18, 19] in which two LTSs are only required to fulfil the bisimulation conditions on a subset B of the collection of actions; transitions labelled by actions not in B are treated as in the standard simulation preorder. Intuitively, one may think of the actions in B as corresponding to the uncontrollable events—see [4, page 4]. The aforementioned paper offers a thorough development of the basic theory of partial bisimulation.

Another recent proposal for a simulation-based behavioural relation over LTSs, called the *covariant-contravariant simulation preorder*, has been put forward in [8], and its theory has been investigated further in [9]. This notion of simulation between LTSs is based on considering a partition of their set of actions into three sets: the collection of covariant actions, that of contravariant actions and the set of bivariant actions. Intuitively, one may think of the covariant actions as being under the control of the specification LTS, and transitions with such actions as their label should be simulated by any correct implementation of the specification. On the other hand, the contravariant actions may be considered as being under the control of the implementation (or of the environment) and transitions with such actions as their label should be simulated by the specification. The bivariant actions are treated as in the classic notion of bisimulation.

It is natural to wonder whether there are any relations among these three formalisms. In particular, one may ask oneself whether it is possible to offer mutual translations between specifications given in those state-transition-based models that preserve, and reflect, the appropriate notions of behavioural preorder as well as properties expressed in the modal logics that accompany them—see, e.g., [4, 6, 9]. The aim of this study is to offer an answer to this question.

In this paper, we study the relationships between refinement over modal transition systems, and the covariant-contravariant simulation and the partial bisimulation preorders over labelled transition systems. We offer mutual trans-

lations between modal transition systems and labelled transition systems that preserve, and reflect, refinement and the covariant-contravariant simulation preorder, as well as the modal properties that can be expressed in the logics that characterize those preorders. We also give a translation from labelled transition systems modulo the partial bisimulation preorder into the same model modulo the covariant-contravariant simulation preorder, together with some evidence that the former model is less expressive than the latter. Finally, in order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, we phrase and study their connections in the context of institutions [12].

The developments in this paper indicate that the formalism of MTSs may be seen as a common ground within which one can embed LTSs modulo the covariant-contravariant simulation preorder or partial bisimilarity. Moreover, there are some interesting, and non-obvious, corollaries that one may infer from the translations we provide. See Section 5, where we use our translations to show, e.g., that checking whether two states in an LTS are related by the covariant-contravariant simulation preorder can always be reduced to an equivalent check in a setting without bivariant actions, and provide a more detailed analysis of the translations. The study of the relative expressive power of different formalisms is, however, an art as well as a science, and may yield different answers depending on the conceptual framework that one adopts for the comparison. For instance, at the level of institutions [12], we provide an institution morphism from the institution corresponding to the theory of MTSs modulo refinement into the institution corresponding to the theory of LTSs modulo the covariant-contravariant simulation preorder. However, we conjecture that there is no institution morphism in the other direction. The work presented in the study opens several interesting avenues for future research, and settling the above conjecture is one of a wealth of research questions we survey in Section 9.

The remainder of the paper is organized as follows. Section 2 is devoted to preliminaries. In particular, in that section, we provide all the necessary background on modal and labelled transition systems, modal refinement and the covariant-contravariant simulation preorder, and the modal logics that characterize those preorders. In Section 3, we show how one can translate LTSs modulo the covariant-contravariant simulation preorder into MTSs modulo refinement. Section 4 presents the converse translation. We discuss the mutual translations between LTSs and MTSs in Section 5. As described in Section 6, the translation from MTSs and their modal logic to the realm of LTSs modulo the covariant-contravariant simulation preorder can be used to transfer the characteristic-formula result from [6] to one for LTSs modulo the covariant-contravariant simulation preorder. Section 7 offers a translation from LTSs modulo partial bisimilarity into LTSs modulo the covariant-contravariant simulation preorder. In Section 8, we study the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder in the context of institutions. Sec-

tion 9 concludes the paper and offers a number of directions for future research that we plan to pursue.

This article is a substantially expanded version of the conference paper [1]. Apart from including the proofs of all the technical results, which were announced without proof in the conference publication with the exception of three propositions, as well as further remarks and explanations, the following contributions are new in this version of the paper:

- the discussion of the translation $\mathcal{MC}$ from Boudol-Larsen modal formulae to covariant-contravariant formulae presented on pages 9–10;
- the discussion of the translation $\mathcal{C}^{-1}$ from covariant-contravariant formulae to Boudol-Larsen modal formulae presented on page 13;
- the material in Section 6; and
- the material on page 28 regarding a conjecture from [1].

2. Preliminaries

We begin by introducing modal transition systems, with their associated notion of (modal) refinement, and labelled transition systems modulo the covariant-contravariant simulation preorder. We refer the reader to, e.g., [6, 16, 17] and [8, 9] for more information, motivation and examples.

2.1. Modal transition systems and refinement

Definition 1. For a set of actions A, a *modal transition* system (MTS) is a triple $(P, \to_\diamond, \to_\Box)$, where P is a set of states and $\to_\diamond, \to_\Box \subseteq P \times A \times P$ are transition relations such that $\to_\Box \subseteq \to_\diamond$.

An MTS is *image finite* iff the set $\{p' \mid p \xrightarrow{a}_\diamond p'\}$ is finite for each $p \in P$ and $a \in A$.

The transitions in $\to_\Box$ are called the *must transitions* and those in $\to_\diamond$ are the *may transitions*. In an MTS, each must transition is also a may transition, which intuitively means that any required transition is also allowed.

In what follows, we often identify an MTS, or a transition system of any of the types that we consider in this paper, with its set of states. In case we wish to make clear the 'ambient' transition system in which a state p lives, we write (P, p) to indicate that p is to be viewed as a state in P.

The notion of (modal) refinement $\sqsubseteq$ over MTSs that we now proceed to introduce is based on the idea that if $p \sqsubseteq q$ then q is a 'refinement' of the specification p. In that case, intuitively, q may be obtained from p by possibly

- removing some of its may transitions and/or
- turning some of its may transitions into must transitions.

Definition 2. A relation $R \subseteq P \times Q$ is a *refinement relation* between two modal transition systems if, whenever $p\ R\ q$:

- $p \xrightarrow{a}_{\Box} p'$ implies that there exists some q' such that $q \xrightarrow{a}_{\Box} q'$ and $p'\ R\ q'$;
- $q \xrightarrow{a}_{\diamond} q'$ implies that there exists some p' such that $p \xrightarrow{a}_{\diamond} p'$ and $p'\ R\ q'$.

We write $\sqsubseteq$ for the largest refinement relation.

Example 1. Consider the MTS U over the set of actions A with u as its only state, and transitions $u \xrightarrow{a}_{\diamond} u$ for each $a \in A$. It is well known, and not hard to see, that $u \sqsubseteq p$ holds for each state p in any MTS over action set A. The state u is often referred to as the *loosest (or universal) specification*.

Definition 3. Given a set of actions A, the collection of *Boudol-Larsen's modal formulae [6]* is given by the following grammar:

$$\varphi ::= \bot \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [a]\varphi \mid \langle a \rangle \varphi \qquad (a \in A).$$

The semantics of these formulae with respect to an MTS P and a state $p \in P$ is defined by means of the satisfaction relation $\models$, which is the least relation satisfying the following clauses:

$(P, p) \models \top$.

$(P, p) \models \varphi_1 \wedge \varphi_2$ if $(P, p) \models \varphi_1$ and $(P, p) \models \varphi_2$.

$(P, p) \models \varphi_1 \vee \varphi_2$ if $(P, p) \models \varphi_1$ or $(P, p) \models \varphi_2$.

$(P, p) \models [a]\varphi$ if $(P, p') \models \varphi$ for all $p \xrightarrow{a}_{\diamond} p'$.

$(P, p) \models \langle a \rangle \varphi$ if $(P, p') \models \varphi$ for some $p \xrightarrow{a}_{\Box} p'$.

We say that a formula is *existential* if it does not contain occurrences of $[a]$-operators, $a \in A$.

For example, the state u in the MTS U from Example 1 satisfies neither the formula $\langle a \rangle \top$ nor the formula $[a]\bot$. Indeed, it is not hard to see that (U, u) satisfies a formula φ if, and only if, φ is a tautology.

The following result stems from [6].

Proposition 1. *Let p, q be states in image-finite MTSs over the set of actions A. Then $p \sqsubseteq q$ iff the collection of Boudol-Larsen's modal formulae satisfied by p is included in the collection of formulae satisfied by q.*

2.2. Labelled transition systems and covariant-contravariant simulation

A labelled transition system (LTS) is just an MTS with $\rightarrow_\Diamond = \rightarrow_\Box$. In what follows, we write $\rightarrow$ for the transition relation in an LTS.

Definition 4. Let P and Q be two LTSs over the set of actions A, and let $\{A^r, A^l, A^{bi}\}$ be a partition of A[1]. An (A^r, A^l)-*simulation* (or just a *covariant-contravariant simulation* when the partition of the set of actions A is understood from the context) between P and Q is a relation $R \subseteq P \times Q$ such that, whenever $p\ R\ q$, we have:

- For all $a \in A^r \cup A^{bi}$ and all $p \xrightarrow{a} p'$, there exists some $q \xrightarrow{a} q'$ with $p'\ R\ q'$.
- For all $a \in A^l \cup A^{bi}$ and all $q \xrightarrow{a} q'$, there exists some $p \xrightarrow{a} p'$ with $p'\ R\ q'$.

We will write $p \lesssim_{cc} q$ if there exists a covariant-contravariant simulation R such that $p\ R\ q$.

The actions in the set A^r are sometimes called *covariant*, those in A^l are *contravariant* and the ones in A^{bi} are *bivariant*. When working with covariant-contravariant simulations, we shall sometimes refer to the triple (A^r, A^l, A^{bi}) as the *signature* of the corresponding LTS, and we will say that such a system is a covariant-contravariant LTS.

Example 2. Assume that $a \in A^r$ and $b \in A^l$. Consider the LTSs described by the CCS [18] terms $p = a + b$, $q = a$ and $r = b$. Then $r \lesssim_{cc} p \lesssim_{cc} q$, but none of the converse relations holds.

Definition 5. *Covariant-contravariant modal logic* has almost the same syntax as the one for modal refinement:

$$\varphi ::= \bot \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [b]\varphi \mid \langle a \rangle \varphi \qquad (a \in A^r \cup A^{bi}, b \in A^l \cup A^{bi}).$$

However, the semantics differs for the modal operators, since we interpret formulae over ordinary LTSs:

$(P, p) \models [b]\varphi$ if $(P, p') \models \varphi$ for all $p \xrightarrow{b} p'$.

$(P, p) \models \langle a \rangle \varphi$ if $(P, p') \models \varphi$ for some $p \xrightarrow{a} p'$.

For example, both p and q from Example 2 satisfy the formula $\langle a \rangle \top$, while r does not. On the other hand, q satisfies the formula $[b]\bot$, but neither p nor r do.

The following result stems from [9].

Proposition 2. *Let p, q be states in image-finite LTSs with the same signature. Then $p \lesssim_{cc} q$ iff the collection of covariant-contravariant modal formulae satisfied by p is included in the collection of covariant-contravariant modal formulae satisfied by q.*

[1] Note that any of the sets A^r, A^l and A^{bi} may be empty. Our use of the word 'partition' is therefore non-standard.

3. From covariant-contravariant simulations to modal refinements

We start our study of the connections between MTSs modulo refinement and LTSs modulo the covariant-contravariant simulation preorder, by showing that LTSs modulo $\lesssim_{cc}$ may be translated into MTSs modulo $\sqsubseteq$. Such a translation preserves, and reflects, those preorders and the satisfaction of modal formulae. This result is, at first, a bit surprising, since covariant-contravariant systems look more expressive than modal systems because they contain three different kinds of actions, which moreover are totally independent from each other, while modal systems only contain two kinds of transitions, which besides are strongly related, since any must transition is also a may one.

Definition 6. Let P be a covariant-contravariant LTS with signature $\{A^r, A^l, A^{bi}\}$. Then the associated MTS $\mathcal{M}(P)$ is constructed as follows:

- The set of actions of $\mathcal{M}(P)$ is $A = A^r \cup A^l \cup A^{bi}$.
- The set of states of $\mathcal{M}(P)$ is that of P plus a new state u.
- For each transition $p \xrightarrow{a} p'$ in P, add a may transition $p \xrightarrow{a}_{\diamond} p'$ in $\mathcal{M}(P)$.
- For each transition $p \xrightarrow{a} p'$ in P with $a \in A^r \cup A^{bi}$, add a must transition $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$.
- For each a in A^r and state p, add the transition $p \xrightarrow{a}_{\diamond} u$ to $\mathcal{M}(P)$, as well as transitions $u \xrightarrow{a}_{\diamond} u$ for each action $a \in A$.
- There are no other transitions in $\mathcal{M}(P)$.

The following proposition essentially states that the translation $\mathcal{M}$ is correct.

Proposition 3. *A relation R is a covariant-contravariant simulation between LTSs P and Q iff $\mathcal{M}(R)$ is a refinement between $\mathcal{M}(P)$ and $\mathcal{M}(Q)$, where $\mathcal{M}(R) = R \cup \{(u, q) \mid q$ a state of $\mathcal{M}(Q)\}$.*

Proof. We prove the two implications separately.

($\Rightarrow$) Assume that R is a covariant-contravariant simulation. We shall prove that $\mathcal{M}(R)$ is a refinement.

Suppose that $p\ R\ q$ and $q \xrightarrow{a}_{\diamond} q'$ in $\mathcal{M}(Q)$. By the definition of $\mathcal{M}(Q)$, the transition $q \xrightarrow{a} q'$ is in Q. If $a \in A^l \cup A^{bi}$, since $p\ R\ q$ and R is a covariant-contravariant simulation, we have that $p \xrightarrow{a} p'$ in P for some p' such that $p'\ R\ q'$. By the construction of $\mathcal{M}(P)$, it holds that $p \xrightarrow{a}_{\diamond} p'$ and we are done. If $a \in A^r$, then $p \xrightarrow{a}_{\diamond} u$ and $u\ \mathcal{M}(R)\ q'$, as required.

Assume now that $p\ R\ q$ and $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$. Then $p \xrightarrow{a} p'$ in P with $a \in A^r \cup A^{bi}$. As R is a covariant-contravariant simulation, it follows that $q \xrightarrow{a} q'$ in Q for some q' such that $p'\ R\ q'$. Since $a \in A^r \cup A^{bi}$, there is a must transition $q \xrightarrow{a}_{\Box} q'$ in $\mathcal{M}(Q)$, and we are done. To finish the proof of this implication, recall that, as shown in Example 1, each state q is a refinement of u.

($\Leftarrow$) Assume that $\mathcal{M}(R)$ is a refinement. We shall prove that R is a covariant-contravariant simulation.

Suppose that $p\ R\ q$ and $q \xrightarrow{a} q'$ in Q with $a \in A^l \cup A^{bi}$. Then $q \xrightarrow{a}_{\diamond} q'$ in $\mathcal{M}(Q)$. Since $\mathcal{M}(R)$ is a refinement, in $\mathcal{M}(P)$ we have that $p \xrightarrow{a}_{\diamond} p'$ for some p' (different from u, because $a \notin A^r$) such that $p'\ R\ q'$. By the construction of $\mathcal{M}(P)$, it follows that $p \xrightarrow{a} p'$ in P and we are done.

Suppose now that $p\ R\ q$ and $p \xrightarrow{a} p'$ in P with $a \in A^r \cup A^{bi}$. Then $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$. Since $\mathcal{M}(R)$ is a refinement, there is some q' (again, different from u) such that $q \xrightarrow{a}_{\Box} q'$ in $\mathcal{M}(Q)$ and $p'\ R\ q'$. By the construction of $\mathcal{M}(Q)$, it follows that $q \xrightarrow{a} q'$ in Q and we are done. □

Remark 1. As witnessed by the proof of the above proposition, the role of the transitions $p \xrightarrow{a}_{\diamond} u$ in $\mathcal{M}(P)$ with $a \in A^r$, where u is the loosest specification from Example 1, is to satisfy 'for free' the proof obligations that are generated, in the setting of modal refinement, by representing A^r-labelled transitions in an LTS P by means of must transitions in $\mathcal{M}(P)$. This is in the spirit of the developments in [15], where the standard simulation preorder is cast in a coalgebraic framework by phrasing it in the setting of bisimilarity. The coalgebraic recasting of simulation as a bisimulation is done in such a way that the added proof obligations that are present in the definition of bisimilarity are automatically satisfied.

Corollary 4. *Let P and Q be two LTSs with the same signature, and let $p \in P$ and $q \in Q$. Then $(P,p) \lesssim_{cc} (Q,q)$ iff $(\mathcal{M}(P),p) \sqsubseteq (\mathcal{M}(Q),q)$.*

Definition 7. Let us extend $\mathcal{M}$ to translate formulae over the modal logic that characterizes the covariant-contravariant simulation preorder to the modal logic for modal transition systems by simply defining $\mathcal{M}(\varphi) = \varphi$.

Proposition 5. *If P is an LTS and φ is a formula of the logic that characterizes covariant-contravariant simulation, then for each $p \in P$:*

$$(P,p) \models \varphi \iff (\mathcal{M}(P),p) \models \mathcal{M}(\varphi).$$

Proof. By structural induction on φ. The only non-trivial cases are the ones corresponding to the modal operators, which we detail below. (In all the following proofs, the steps labelled 'IH' are those that use the induction hypothesis.)

- $\langle a\rangle\varphi$, with $a \in A^r \cup A^{bi}$.

$$\begin{array}{lcl}
(P,p) \models \langle a\rangle\varphi & \iff & \text{there is } p \xrightarrow{a} p' \text{ in } P \text{ with } (P,p') \models \varphi \\
& \overset{\text{IH}}{\iff} & \text{there is } p \xrightarrow{a}_{\Box} p' \text{ in } \mathcal{M}(P) \text{ with } (\mathcal{M}(P),p') \models \mathcal{M}(\varphi) \\
& \iff & (\mathcal{M}(P),p) \models \langle a\rangle\mathcal{M}(\varphi) \\
& \iff & (\mathcal{M}(P),p) \models \mathcal{M}(\langle a\rangle\varphi)
\end{array}$$

- $[a]\varphi$, with $a \in A^l \cup A^{bi}$.

$$\begin{array}{lll}
(P,p) \models [a]\varphi & \iff & (P,p') \models \varphi \text{ for all } p \xrightarrow{a} p' \text{ in } P \\
& \overset{\text{IH}}{\iff} & (\mathcal{M}(P),p') \models \mathcal{M}(\varphi) \text{ for all } p \xrightarrow{a}_{\diamond} p' \text{ in } \mathcal{M}(P) \\
& & (\text{note that } p \xrightarrow{a}_{\diamond} u \text{ only for } a \in A^r) \\
& \iff & (\mathcal{M}(P),p) \models [a]\mathcal{M}(\varphi) \\
& \iff & (\mathcal{M}(P),p) \models \mathcal{M}([a]\varphi)
\end{array}$$

□

It is natural to wonder whether it is possible to provide a version of Proposition 5 for formulae in Boudol-Larsen modal logic. In particular, it would be interesting to characterize the collections of formulae in Boudol-Larsen modal logic whose satisfaction is preserved by $\mathcal{M}$, in a suitable technical sense. In order to address this question, let $\{A^r, A^l, A^{bi}\}$ be the signature of some LTS P and let $A = A^r \cup A^l \cup A^{bi}$.

Define the transformation $\mathcal{MC}$ from Boudol-Larsen formulae over A to covariant-contravariant formulae over the signature $\{A^r, A^l, A^{bi}\}$ as follows:

Definition 8. $\mathcal{MC}$ is the unique homomorphism satisfying:

- $\mathcal{MC}(\langle a\rangle\varphi) = \begin{cases} \langle a\rangle\mathcal{MC}(\varphi) & \text{if } a \in A^r \cup A^{bi} \\ \bot & \text{otherwise} \end{cases}$
- $\mathcal{MC}([a]\varphi) = \begin{cases} [a]\mathcal{MC}(\varphi) & \text{if } a \in A^l \cup A^{bi} \\ \top & \text{otherwise} \end{cases}$

The interplay between the transformation function $\mathcal{M}$ between LTSs and MTSs, and the function $\mathcal{MC}$ operating on Boudol-Larsen formulae is fully described by the following results.

Proposition 6. *Let P an LTS over signature $\{A^r, A^l, A^{bi}\}$ and let $p \in P$. Suppose that φ is a formula in Boudol-Larsen modal logic over $A = A^r \cup A^l \cup A^{bi}$. Then the following statements hold:*

1. *If $(\mathcal{M}(P),p) \models \varphi$ then $(P,p) \models \mathcal{MC}(\varphi)$.*
2. *If $(P,p) \models \mathcal{MC}(\varphi)$ and (either φ is existential or $A^r = \emptyset$) then $(\mathcal{M}(P),p) \models \varphi$.*

Proof. We prove the two statements separately.

1. We proceed by induction on the structure of φ and focus on the cases involving the modal operators.

- Case $\varphi = \langle a \rangle \varphi'$. Assume that $(\mathcal{M}(P), p) \models \langle a \rangle \varphi'$. This means that there is some p' such that $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$ and $(\mathcal{M}(P), p') \models \varphi'$. By the definition of $\mathcal{M}$, $a \in A^r \cup A^{bi}$ and $p \xrightarrow{a} p'$. Moreover, by the inductive hypothesis, $(P, p) \models \mathcal{MC}(\varphi')$. Therefore, $(P, p) \models \langle a \rangle \mathcal{MC}(\varphi')$, and since $a \in A^r \cup A^{bi}$, $(P, p) \models \mathcal{MC}(\langle a \rangle \varphi')$.
- Case $\varphi = [a]\varphi'$. Assume that $(\mathcal{M}(P), p) \models [a]\varphi'$. If $a \in A^r$ then there is nothing to prove, since $\mathcal{MC}(\varphi) = \top$. Assume therefore that $a \in A^l \cup A^{bi}$. We will prove that $(p, p) \models [a]\mathcal{MC}(\varphi')$. To this end, suppose that $p \xrightarrow{a} p'$ in P with $a \in A^l \cup A^{bi}$. By the definition of $\mathcal{M}$ we have that $p \xrightarrow{a}_{\diamond} p'$ in $\mathcal{M}(P)$. Since $(\mathcal{M}(P), p) \models [a]\varphi'$, it follows that $(\mathcal{M}(P), p') \models \varphi'$. The Inductive hypothesis yields $(P, p') \models \mathcal{MC}(\varphi')$, which was to be shown.

2. Assume that $(P, p) \models \mathcal{MC}(\varphi)$ and (either φ is existential or $A^r = \emptyset$). We show that $(\mathcal{M}(P), p) \models \varphi$ by induction on the structure of φ. Again, the only interesting cases are those dealing with the modal operators.
   - Case $\varphi = \langle a \rangle \varphi'$. Since $(P, p) \models \mathcal{MC}(\varphi)$, we have that $a \in A^r \cup A^{bi}$ and that $p \xrightarrow{a} p'$ for some p' such that $(P, p') \models \mathcal{MC}(\varphi')$. Since φ' is either existential or $A^r = \emptyset$, we may apply the inductive hypothesis to infer that $(\mathcal{M}(P), p') \models \varphi'$. By the definition of $\mathcal{M}$, we have that $p \xrightarrow{a}_m ustp'$. Therefore $(\mathcal{M}(P), p) \models \langle a \rangle \varphi'$, which was to be shown.
   - Case $\varphi = [a]\varphi'$. Since φ is not existential, we have that $A^r = \emptyset$. So $\mathcal{MC}([a]\varphi') = [a]\mathcal{MC}(\varphi')$ and $(P, p) \models [a]\mathcal{MC}(\varphi')$ by assumption.
     Let $p \xrightarrow{a}_{\diamond} p'$ in $\mathcal{M}(P)$. As $a \in A^r \cup A^{bi}$, it follows that $p \xrightarrow{a} p'$ in P. Therefore $(P, p') \models \mathcal{MC}(\varphi')$. By induction, $(\mathcal{M}(P), p') \models \varphi'$. Since $p \xrightarrow{a}_{\diamond} p'$ was chosen arbitrary, it follows that $(\mathcal{M}(P), p) \models [a]\varphi'$, and we are done. □

Remark 2. The proviso that φ is existential or $A^r = \emptyset$ is necessary in statement 2 of the above proposition. To see this, assume that $a \in A^r$ and consider the Boudol-Larsen formula $[a]\bot$. Then the LTS described by the covariant-contravariant system term 0 satisfies $\top = \mathcal{MC}([a]\bot)$. On the other hand, $0 \xrightarrow{a}_{\diamond} u$ holds in $\mathcal{M}(0)$, and therefore $(\mathcal{M}(0), 0) \not\models [a]\bot$. This point is related to some observations we shall present in Section 8.

Remark 3 (Open question). Is there a (compositional) translation from Boudol-Larsen logic to covariant-contravariant modal logic such that

$$(P, p) \models \varphi \text{ implies } (\mathcal{M}(P), p) \models \varphi$$

for all LTSs P, states $p \in P$ and formulae φ?

4. From modal refinements to covariant-contravariant simulations

We next show that MTSs modulo $\sqsubseteq$ may be translated into LTSs modulo $\lesssim_{cc}$. As the one studied in the previous section, our translation preserves, and reflects, those preorders and the satisfaction of modal formulae. This is, to our mind, a less surprising result than the one presented in the previous section, even if in order to obtain it we have to introduce two "copies" of each action $a \in A$: one covariant $cv(a) \in A^r$ to represent must transitions, and another contravariant $ct(a) \in A^l$ to represent may transitions. As a matter of fact, we do not need the additional generality that is offered by the possibility of also having bivariant actions in the signature to adequately represent any MTS.

Definition 9. Let M be an MTS with set of actions A. The LTS $\mathcal{C}(M)$, with signature $A^r = \{\mathrm{cv}(a) \mid a \in A\}$, $A^l = \{\mathrm{ct}(a) \mid a \in A\}$ and $A^{bi} = \emptyset$, is constructed as follows:

- The set of states of $\mathcal{C}(M)$ is the same as that of M.
- For each transition $p \xrightarrow{a}_{\diamond} p'$ in M, add $p \xrightarrow{\mathrm{ct}(a)} p'$ to $\mathcal{C}(M)$.
- For each transition $p \xrightarrow{a}_{\Box} p'$ in M, add $p \xrightarrow{\mathrm{cv}(a)} p'$ to $\mathcal{C}(M)$.
- There are no other transitions in $\mathcal{C}(M)$.

Observe that the LTSs obtained as a translation of an MTS have the following properties:

1. $A^{bi} = \emptyset$ and
2. there is a bijection $h : A^r \to A^l$ such that if $p \xrightarrow{a} p'$ with $a \in A^r$ then $p \xrightarrow{h(a)} p'$.

The latter requirement corresponds to the fact that each must transition in an MTS is also a may transition.

The following proposition states that the translation $\mathcal{C}$ is correct.

Proposition 7. *A relation R is a refinement between P and Q iff R is a covariant-contravariant simulation between $\mathcal{C}(P)$ and $\mathcal{C}(Q)$.*

Proof. We prove the two implications separately.

($\Rightarrow$) Assume that $p \mathrel{R} q$. If $p \xrightarrow{\mathrm{cv}(a)} p'$ in $\mathcal{C}(P)$ then, by construction, $p \xrightarrow{a}_{\Box} p'$ in P. Since R is a refinement, there is some q' in Q with $q \xrightarrow{a}_{\Box} q'$ and $p' \mathrel{R} q'$. Since $q \xrightarrow{\mathrm{cv}(a)} q'$ is in $\mathcal{C}(Q)$ by construction, we are done. Now, assume that $q \xrightarrow{\mathrm{ct}(a)} q'$ in $\mathcal{C}(Q)$. Then $q \xrightarrow{a}_{\diamond} q'$ in Q and, since R is a refinement, $p \xrightarrow{a}_{\diamond} p'$ in P for some p' with $p' \mathrel{R} q'$. By construction, $p \xrightarrow{\mathrm{ct}(a)} p'$ is in $\mathcal{C}(P)$ and we are done.

($\Leftarrow$) Assume that $p \mathrel{R} q$. If $q \xrightarrow{a}_{\diamond} q'$ in Q then $q \xrightarrow{\mathrm{ct}(a)} q'$ in $\mathcal{C}(Q)$ and, since R is a covariant-contravariant simulation, $p \xrightarrow{\mathrm{ct}(a)} p'$ for some p' in $\mathcal{C}(P)$ such that

$p' R q'$; hence $p \xrightarrow{a}_{\diamond} p'$ in P as required. Now, if $p \xrightarrow{a}_{\Box} p'$ in P then $p \overset{\mathrm{cv}(a)}{\rightarrow} p'$ in $\mathcal{C}(P)$. Since R is a covariant-contravariant simulation, there is some q' in $\mathcal{C}(Q)$ with $q \overset{\mathrm{cv}(a)}{\rightarrow} q'$ and $p'\ R\ q'$, and therefore $q \xrightarrow{a}_{\Box} q'$ in Q. □

Corollary 8. *Let P and Q be two MTSs with the same action set, and let $p \in P$ and $q \in Q$. Then $(P,p) \sqsubseteq (Q,q)$ iff $(\mathcal{C}(P),p) \lesssim_{cc} (\mathcal{C}(Q),q)$.*

Remark 4. It is easy to see that the mapping $\mathcal{C}$ is injective. Therefore, given an LTS P that is in the range of $\mathcal{C}$, we may write $\mathcal{C}^{-1}(P)$ for the unique MTS whose $\mathcal{C}$-image is P.

Again, we can also extend the translation $\mathcal{C}$ to also translate modal formulae. However, in this case, the change of alphabet requires a simple, but non-trivial, definition of the extension.

Definition 10. Let us extend $\mathcal{C}$ to translate formulae over the modal logic for modal transition systems with set of actions A to the modal logic that characterizes covariant-contravariant simulation with signature $A^r = \{\mathrm{cv}(a) \mid a \in A\}$, $A^l = \{\mathrm{ct}(a) \mid a \in A\}$ and $A^{bi} = \emptyset$.

- $\mathcal{C}(\bot) = \bot$.
- $\mathcal{C}(\top) = \top$.
- $\mathcal{C}(\varphi \wedge \psi) = \mathcal{C}(\varphi) \wedge \mathcal{C}(\psi)$.
- $\mathcal{C}(\varphi \vee \psi) = \mathcal{C}(\varphi) \vee \mathcal{C}(\psi)$.
- $\mathcal{C}(\langle a \rangle \varphi) = \langle \mathrm{cv}(a) \rangle \mathcal{C}(\varphi)$.
- $\mathcal{C}([a]\varphi) = [\mathrm{ct}(a)]\mathcal{C}(\varphi)$.

Proposition 9. *If P is an MTS and φ is a Boudol-Larsen modal formula, then for each $p \in P$:*

$$(P,p) \models \varphi \iff (\mathcal{C}(P),p) \models \mathcal{C}(\varphi).$$

Proof. By structural induction on φ, with the only non-trivial cases being those that correspond to the modal operators:

- $[a]\varphi$, with $a \in A$.

$$\begin{aligned}
(P,p) \models [a]\varphi \quad &\iff \quad (P,p') \models \varphi \text{ for all } p \xrightarrow{a}_{\diamond} p' \text{ in } P \\
&\overset{\text{IH}}{\iff} \quad (\mathcal{C}(P),p') \models \mathcal{C}(\varphi) \text{ for all } p \overset{\mathrm{ct}(a)}{\rightarrow} p' \text{ in } \mathcal{C}(P) \\
&\iff \quad (\mathcal{C}(P),p) \models [\mathrm{ct}(a)]\mathcal{C}(\varphi)
\end{aligned}$$

- $\langle a \rangle \varphi$, with $a \in A$.

$$\begin{array}{lcl} (P,p) \models \langle a \rangle \varphi & \iff & (P,p') \models \varphi \text{ for some } p \stackrel{a}{\rightarrow}_{\Box} p' \text{ in } P \\ & \stackrel{\text{IH}}{\iff} & (\mathcal{C}(P),p') \models \mathcal{C}(\varphi) \text{ for some } p \stackrel{\text{cv}(a)}{\rightarrow} p' \text{ in } \mathcal{C}(P) \\ & \iff & (\mathcal{C}(P),p) \models \langle \text{cv}(a) \rangle \mathcal{C}(\varphi) \end{array}$$

□

Remark 5. In fact, it is very easy to see that the translations $\mathcal{M}$ and $\mathcal{C}$ also preserve, and reflect, the satisfaction of formulae in the extensions of the logics from Definitions 3 and 5 with infinite conjunctions and disjunctions.

It is natural to wonder whether it is possible to provide a version of Proposition 9 for formulae in covariant-contravariant modal logic over the signature $A^r = \{\text{cv}(a) \mid a \in A\}$, $A^l = \{\text{ct}(a) \mid a \in A\}$ and $A^{bi} = \emptyset$. To this end, let $\mathcal{C}^{-1}$ denote the inverse of $\mathcal{C}$ over Boudol-Larsen modal formulae defined in the obvious way. We then have that:

Proposition 10. *Let P be an MTS over the set of actions A, and let φ be a covariant-contravariant modal formula over the signature $A^r = \{\text{cv}(a) \mid a \in A\}$, $A^l = \{\text{ct}(a) \mid a \in A\}$ and $A^{bi} = \emptyset$. Then, for each $p \in P$.*

$$(P,p) \models \mathcal{C}^{-1}(\varphi) \iff (\mathcal{C}(P),p) \models \varphi.$$

Proof. By Proposition 9,

$$(P,p) \models \mathcal{C}^{-1}(\varphi) \iff (\mathcal{C}(P),p) \models \mathcal{C}(\mathcal{C}^{-1}(\varphi)).$$

The claim now follows since $\mathcal{C}(\mathcal{C}^{-1}(\varphi)) = \varphi$. □

The above observation is in contrast with the result we established earlier in Proposition 6. This may be taken to be a first indication that the translation from MTSs to LTSs, and the accompanying one for the associated modal logics, is "more natural" than the one from LTSs to MTSs provided in Section 3. We will explore this issue in more detail in Section 8.

5. Discussion of the previous translations

In Sections 3–4, we saw that it is possible to translate back and forth between the world of LTSs modulo the covariant-contravariant simulation preorder and MTSs modulo refinement. The translations we have presented preserve, and reflect, the preorders and the relevant modal formulae. There are, however, some interesting, and non-obvious, corollaries that one may infer from the translations.

To begin with, assume that P and Q are two LTSs with the same signature, with $A^{bi} \neq \emptyset$. Let $p \in P$ and $q \in Q$ be such that $(P,p) \lesssim_{cc} (Q,q)$. By Corollary 4, we know that this holds exactly when $(\mathcal{M}(P),p) \sqsubseteq (\mathcal{M}(Q),q)$.

Using Corollary 8, we therefore have that checking whether $(P,p) \lesssim_{cc} (Q,q)$ is equivalent to verifying whether $(\mathcal{C}(\mathcal{M}(P)),p) \lesssim_{cc} (\mathcal{C}(\mathcal{M}(Q)),q)$. Note now that A^{bi} is empty in the signature for the LTSs $\mathcal{C}(\mathcal{M}(P))$ and $\mathcal{C}(\mathcal{M}(Q))$. Therefore, checking whether two states are related by the covariant-contravariant simulation preorder can always be reduced to an equivalent check in a setting without bivariant actions.

It is also natural to wonder whether there is any relation between a state p in an LTS P and the equally-named state in $\mathcal{C}(\mathcal{M}(P))$. Similarly, one may wonder whether there is any relation between a state p in an MTS P and the equally-named state in $\mathcal{M}(\mathcal{C}(P))$. In both cases, we are faced with the difficulty arising from the fact that the transition systems resulting from the compositions of the two translations are over the alphabet $\{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\}$, whereas the original system P had transitions labelled by actions in A. In order to overcome this difficulty, we consider the renaming $\rho : \{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\} \to A$ that maps both $\mathrm{cv}(a)$ and $\mathrm{ct}(a)$ to a, for each $a \in A$. Besides, for any transition system P over the set of actions $\{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\}$, we write $\rho(P)$ for the transition system that is obtained from P by renaming the label of each transition in P as indicated by ρ. Then we have the following proposition:

Proposition 11.

1. *Let P be an MTS and $p \in P$. Then we have $(\rho(\mathcal{M}(\mathcal{C}(P))),p) \sqsubseteq (P,p)$.*
2. *Let P be an LTS and $p \in P$. Then we have $(P,p) \lesssim_{cc} (\rho(\mathcal{C}(\mathcal{M}(P))),p)$.*
3. *In general, $(P,p) \sqsubseteq (\rho(\mathcal{M}(\mathcal{C}(P))),p)$ does not hold for an arbitrary MTS P and any state $p \in P$; nor does $(\rho(\mathcal{C}(\mathcal{M}(P))),p) \lesssim_{cc} (P,p)$, for an arbitrary LTS P and any state $p \in P$.*

Proof. We limit ourselves to detailing the proof for the second statement and to offering counter-examples proving the third one. The proof of the first claim follows similar lines to the one for the second, and in fact is even simpler.

In order to prove the second claim, it suffices to show that the identity relation over P is a covariant-contravariant simulation between P and $\rho(\mathcal{C}(\mathcal{M}(P)))$. To this end, assume first that $p \xrightarrow{a} p'$ in P for some $a \in A^r \cup A^{bi}$. Then $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$. Therefore, $p \xrightarrow{\mathrm{cv}(a)} p'$ in $\mathcal{C}(\mathcal{M}(P))$ and $p \xrightarrow{a} p'$ in $\rho(\mathcal{C}(\mathcal{M}(P)))$.

Assume now that $p \xrightarrow{a} p'$ in $\rho(\mathcal{C}(\mathcal{M}(P)))$ for some $a \in A^l \cup A^{bi}$. This means that either $p \xrightarrow{\mathrm{cv}(a)} p'$ or $p \xrightarrow{\mathrm{ct}(a)} p'$ in $\mathcal{C}(\mathcal{M}(P))$. We consider these two possibilities separately.

- Suppose that $p \xrightarrow{\mathrm{cv}(a)} p'$ in $\mathcal{C}(\mathcal{M}(P))$. Then $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{M}(P)$. This means that $p \xrightarrow{a} p'$ in P and $a \in A^r \cup A^{bi}$. By our assumption, it must be the case that $a \in A^{bi}$, and we are done.

- Suppose that $p \xrightarrow{\mathrm{ct}(a)} p'$ in $\mathcal{C}(\mathcal{M}(P))$. Then $p \xrightarrow{a}_{\Diamond} p'$ in $\mathcal{M}(P)$. Since $a \in A^l \cup A^{bi}$ by our assumption, we have that $p' \neq u$ in $\mathcal{M}(P)$, because u can only be reached via A^r-labelled may transitions. Therefore, $p' \in P$ and $p \xrightarrow{a} p'$.

This completes the proof of the second claim.

We now argue that, in general, $(P,p) \sqsubseteq (\rho(\mathcal{M}(\mathcal{C}(P))),p)$ does not hold for an MTS P and a state $p \in P$. Let P be the MTS over the alphabet $A = \{a\}$, with p as its only state and with no transitions. State p has an outgoing a-labelled may transition in $\rho(\mathcal{M}(\mathcal{C}(P)))$, which cannot be matched by p in P. Therefore, $(P,p) \not\sqsubseteq (\rho(\mathcal{M}(\mathcal{C}(P))),p)$.

To complete the proof we now argue that, in general, $(\rho(\mathcal{C}(\mathcal{M}(P))),p) \lesssim_{cc} (P,p)$ does not hold for an LTS P and a state $p \in P$. Let P be an LTS with $A^r = \{a\}$, p as its only state, and with no transitions. The sets A^l and A^{bi} can be arbitrary and play no role in the counter-example. Then it is immediate to see that state p has a transition $p \stackrel{a}{\to} u$ in $\rho(\mathcal{C}(\mathcal{M}(P)))$, but this transition cannot be matched by p in P. □

We shall now present a result on the relationships between the translations $\mathcal{M}$ and $\mathcal{C}$ for LTSs without bivariant actions.

Definition 11. Let P be an LTS with its alphabet partitioned into A^r and A^l. Then the LTS $\overline{P}$ is that obtained from P by simply renaming every $a \in A^r$ as $\mathrm{cv}(a)$ and every $a \in A^l$ as $\mathrm{ct}(a)$.

Proposition 12. *Let P be an LTS over an alphabet $A^r \cup A^l$ and let Q be an MTS over the same alphabet. Then the following statements hold.*

1. *If a relation R is a covariant-contravariant simulation between $\overline{P}$ and $\mathcal{C}(Q)$, then R is a refinement between $\mathcal{M}(P)$ and Q.*
2. *If $(\overline{P},p) \lesssim_{cc} (\mathcal{C}(Q),q)$ then $(\mathcal{M}(P),p) \sqsubseteq (Q,q)$, for all states $p \in P$ and $q \in Q$.*
3. *The converse implication of the above statement fails.*

Proof. We limit ourselves to detailing a proof of the first statement and to offering a counter-example showing the third. The second statement is an immediate corollary of the first.

To prove the first statement, assume that $p\ R\ q$ and that R is a covariant-contravariant simulation between $\overline{P}$ and $\mathcal{C}(Q)$. If $q \stackrel{a}{\to}_\diamond q'$ in Q then $q \stackrel{\mathrm{ct}(a)}{\to} q'$ in $\mathcal{C}(Q)$. Since R is a covariant-contravariant simulation between $\overline{P}$ and $\mathcal{C}(Q)$, there is some p' in $\overline{P}$ with $p \stackrel{\mathrm{ct}(a)}{\to} p'$ and $p'\ R\ q'$. Therefore, $p \stackrel{a}{\to} p'$ in P with $a \in A^l$, and $p \stackrel{a}{\to}_\diamond p'$ in $\mathcal{M}(P)$ with $p'\ R\ q'$, as required. Now, if $p \stackrel{a}{\to}_\Box p'$ in $\mathcal{M}(P)$ then $p \stackrel{a}{\to} p'$ in P with $a \in A^r$ and $p \stackrel{\mathrm{cv}(a)}{\to} p'$ in $\overline{P}$. Since R is a covariant-contravariant simulation between $\overline{P}$ and $\mathcal{C}(Q)$, there is some q' in $\mathcal{C}(Q)$ with $q \stackrel{\mathrm{cv}(a)}{\to} q'$ and $p'\ R\ q'$, and therefore $q \stackrel{a}{\to}_\Box q'$ in Q, as required.

To see that the converse implication of the second statement in the proposition fails in general, let P be an LTS with $A^r = \{a\}$, with p as its only state and with no transitions. In this case A^l can be arbitrary and plays no role in the counter-example. Let Q be a one-state MTS with the transition $q \stackrel{a}{\to}_\diamond q$. Then we have $(\mathcal{M}(P),p) \sqsubseteq (Q,q)$. On the other hand, $(\overline{P},p) \not\lesssim_{cc} (\mathcal{C}(Q),q)$, because $q \stackrel{\mathrm{ct}(a)}{\to} q$ in $\mathcal{C}(Q)$ and $\mathrm{ct}(a)$ is a contravariant action, whereas the LTS $\overline{P}$ has no transitions. □

6. Characteristic formulae for processes

In this section, we show that the translation $\mathcal{C}$ can be used to transfer characteristic formulae from the setting of MTSs modulo refinement to that of LTSs modulo the covariant-contravariant simulation preorder. For consistency with the developments in [6], we focus on characteristic formulae for finite, "essentially loop-free" structures. Following [6, 9], we consider two signatures: the first generates terms describing a family of MTSs, and the second generates terms denoting a family of LTSs.

Definition 12 ([6]). Given a set of actions A, the set $\mathcal{T}_M(A)$ of MTS process terms is given by

$$t ::= 0 \mid \omega \mid a.t \mid a!t \mid t + t.$$

where $a \in A$.

We define the 'universal MTS' associated with $\mathcal{T}_M(A)$ as follows:

- Its set of states is just $\mathcal{T}_M(A)$.
- For each term $a.t$ we have the transition $a.t \xrightarrow{a}_{\diamond} t$; besides, for each $a \in A$, we have $\omega \xrightarrow{a}_{\diamond} \omega$.
- For each term $a!t$, and $\mathsf{o} \in \{\Box, \diamond\}$ we have $a!t \xrightarrow{a}_{\mathsf{o}} t$.
- For each term $t_1 + t_2$, $a \in A$ and $\mathsf{o} \in \{\Box, \diamond\}$ we have $t_1 + t_2 \xrightarrow{a}_{\mathsf{o}} t'$, if and only if, we have $t_i \xrightarrow{a}_{\mathsf{o}} t'$ for some $i \in \{1, 2\}$.

Note that ω denotes the MTS U from Example 1 and is the only source of loops in the MTS we have just described. So, abstracting from the self-loops at the leaves labelled with ω, terms in $\mathcal{T}_M(A)$ may be viewed as describing finite synchronization trees, in the sense of Milner.

Definition 13. Let $(A^r, A^l, \emptyset)$ be a signature and let $A = A^r \cup A^l$. The set $\mathcal{T}_L(A)$ of LTS process terms is given by

$$t ::= 0 \mid \omega \mid a.t \mid t + t,$$

where $a \in A$.

We define the 'universal LTS' associated with $\mathcal{T}_L(A)$ as follows:

- Its set of states is just $\mathcal{T}_L(A)$.
- For each term at we have the transition $at \xrightarrow{a} t$; besides, for each $a \in A^l$, we have $\omega \xrightarrow{a} \omega$.
- For each term $t_1 + t_2$ and each $a \in A$, we have $t_1 + t_2 \xrightarrow{a} t'$, if and only if, we have $t_i \xrightarrow{a} t'$ for some $i \in \{1, 2\}$.

The translation $\mathcal{C}$ from MTSs over the alphabet A to LTSs over the signature $(\{\mathrm{cv}(a) \mid a \in A\}, \{\mathrm{ct}(a) \mid a \in A\}, \emptyset)$ can be extended to terms in $\mathcal{T}_M(A)$ yielding terms in $\mathcal{T}_L(\{\mathrm{cv}(a), \mathrm{ct}(a) \mid a \in A\})$ as the unique homomorphism that is the identity over constants and satisfies the following equalities:

$$\begin{aligned} \mathcal{C}(a!t) &= \mathrm{cv}(a).\mathcal{C}(t) + \mathrm{ct}(a).\mathcal{C}(t) \quad \text{and} \\ \mathcal{C}(a.t) &= \mathrm{ct}(a).\mathcal{C}(t). \end{aligned}$$

Then we have the following results:

Lemma 13. *Let t be an MTS term. Then the following statements hold:*

1. *If $t \xrightarrow{a}_{\Box} t'$ for some MTS term t' then $\mathcal{C}(t) \overset{\mathrm{cv}(a)}{\rightarrow} \mathcal{C}(t')$.*
2. *If $t \xrightarrow{a}_{\diamond} t'$ for some MTS term t' then $\mathcal{C}(t) \overset{\mathrm{ct}(a)}{\rightarrow} \mathcal{C}(t')$.*
3. *If $\mathcal{C}(t) \overset{\mathrm{cv}(a)}{\rightarrow} u$ for some LTS term u then $t \xrightarrow{a}_{\Box} t'$ for some MTS term t' such that $u = \mathcal{C}(t')$.*
4. *If $\mathcal{C}(t) \overset{\mathrm{ct}(a)}{\rightarrow} u$ for some LTS term u then $t \xrightarrow{a}_{\diamond} t'$ for some MTS term t' such that $u = \mathcal{C}(t')$.*

Proof. The first two statements can be proven by induction on the proof of the relevant transition. The third and the fourth statement can be easily shown by induction on the structure of t. □

It is not hard to see that the LTS associated with $\mathcal{C}(t)$, where t is an MTS term, is the LTS one obtains by considering the MTS for term t, defined as in Definition 12, and applying the translation $\mathcal{C}$ from Definition 9 to it. Therefore, the following result follows essentially from Proposition 9. (One can also give a simple proof of this result using Lemma 13 above.)

Proposition 14. *For an MTS term t and a modal formula φ,*

$$t \models \varphi \iff \mathcal{C}(t) \models \mathcal{C}(\varphi).$$

The above result can be used to transfer characteristic formulae for MTS terms modulo refinement to characteristic formulae for their image LTS terms via $\mathcal{C}$.

We begin by recalling the definition of characteristic formulae for MTS terms modulo refinement from [6, 16].

Definition 14 ([6, 16]). For each term $t \in \mathcal{T}_M(A)$, the characteristic formula $\chi(t)$ is defined as follows:

$$\chi(t) = \bigwedge_{\phi \in \delta(t)} \phi \wedge \bigwedge_{a \in A} [a]\gamma_a(t), \tag{1}$$

where the set of formulae $\delta(t)$ and the formulae $\gamma_a(t)$ are given inductively thus

1. $\delta(0) = \emptyset$ and $\gamma_a(0) = \bot$,
2. $\delta(\omega) = \emptyset$ and $\gamma_a(\omega) = \top$,
3. $\delta(a.t) = \emptyset$, $\gamma_a(a.t) = \gamma_a(t)$ and $\gamma_b(a.t) = \bot$ $(b \neq a)$,
4. $\delta(a!t) = \{\langle a\rangle\chi(t)\}$ and $\gamma_b(a!t) = \gamma_b(a.t)$, for each $b \in A$, and
5. $\delta(t_1 + t_2) = \delta(t_1) \cup \delta(t_2)$ and $\gamma_a(t_1 + t_2) = \gamma_a(t_1) \vee \gamma_a(t_2)$.

As usual, an empty conjunction stands for $\top$.

The correctness of the above construction was proved by Larsen in [16].

Proposition 15. *Let $t, t' \in \mathcal{T}_M(A)$. Then $t \sqsubseteq t'$ iff $t' \models \chi(t)$.*

Note that the formula $\chi(\omega)$ is logically equivalent to $\top$. Moreover, for each term $t \in \mathcal{T}_M(A)$, we have that, up to logical equivalence,

$$\bigwedge_{\phi \in \delta(t)} \phi = \bigwedge\{\langle a\rangle\chi(t') \mid t \xrightarrow{a}_{\Box} t'\}.$$

Consider now the second conjunction in the formula (1). If t can perform an a-labelled may transition leading to a term that is equivalent to ω with respect to the kernel of $\sqsubseteq$, then, up to logical equivalence,

$$[a]\gamma_a(t) = \top.$$

For each term t, let A_t be the subset of A consisting of all the actions a such that each a-labelled may transition from t leads to a term that is *not* equivalent to ω with respect to the kernel of $\sqsubseteq$. Then, up to logical equivalence,

$$\bigwedge_{a \in A} [a]\gamma_a(t) = \bigwedge_{a \in A_t} [a] \bigvee\{\chi(t') \mid t \xrightarrow{a}_{\diamond} t'\}.$$

In summary, working up to logical equivalence, we can rewrite the formula (1) thus:

$$\bigwedge\{\langle a\rangle\chi(t') \mid t \xrightarrow{a}_{\Box} t'\} \vee \bigwedge_{a \in A_t} [a] \bigvee\{\chi(t') \mid t \xrightarrow{a}_{\diamond} t'\}.$$

Proposition 16. *$\mathcal{C}(\chi(t))$ is a characteristic formula for $\mathcal{C}(t)$, for each $t \in \mathcal{T}_M(A)$.*

Proof. By Propositions 15 and 14, $\mathcal{C}(t) \models \mathcal{C}(\chi(t))$. Now, assume that $s \models \mathcal{C}(\chi(t))$ for some $s \in \mathcal{T}_L(\{\mathrm{cv}(a), ct(a) \mid a \in A\})$. We shall show that $\mathcal{C}(t) \lesssim_{cc} s$. (Observe, in passing, that, since the map $\mathcal{C}$ is not surjective, the term s might not be the image of any MTS term.) To this end, it suffices to show that the relation

$$R = \{(\mathcal{C}(t), s) \mid s \models \mathcal{C}(\chi(t)), s \in \mathcal{T}_L(\{\mathrm{cv}(a), ct(a) \mid a \in A\}), t \in \mathcal{T}_M(A)\}$$

is a covariant-contravariant simulation.

To see this, note, first of all, that, in the light of the above discussion,

$$\mathcal{C}(\chi(t)) = \bigwedge\{\langle \mathrm{cv}(a)\rangle \mathcal{C}(\chi(t')) \mid t \xrightarrow{a}_{\Box} t'\} \vee \bigwedge_{a\in A_t} [\mathrm{ct}(a)] \bigvee\{\mathcal{C}(\chi(t')) \mid t \xrightarrow{a}_{\diamond} t'\}.$$

The claim can now be easily shown using Lemma 13 and the fact that $\mathcal{C}(\omega) = \omega \lesssim_{cc} s'$, for each $s' \in \mathcal{T}_L(\{\mathrm{cv}(a), ct(a) \mid a \in A\})$. □

This last result can be used as an alternative to [2, Lemma 2] to prove the existence of characteristic formulae for LTS terms that are in the range of $\mathcal{C}$. Indeed, for those terms, the characteristic formula derived using the above proposition coincides with the one offered by the direct construction given in the above-cited reference.

7. Partial bisimulation

The partial bisimulation preorder has been proposed in [4] as a suitable behavioural relation over LTSs for studying the theory of supervisory control [20] in a concurrency-theoretic framework. Formally, the notion of partial bisimulation is defined over LTSs with a set of actions A and a so-called *bisimulation set* $B \subseteq A$. The LTSs considered in [4] also include a termination predicate $\downarrow$ over states. For the sake of simplicity, and since its role is orthogonal to our aims in this paper, instead of extending MTSs and their refinements and/or covariant-contravariant simulations with such a predicate, we simply omit this predicate in what follows.

Definition 15. A *partial bisimulation with bisimulation set* B between two LTSs P and Q is a relation $R \subseteq P \times Q$ such that, whenever $p\ R\ q$:

- For all $a \in A$, if $p \xrightarrow{a} p'$ then there exists some $q \xrightarrow{a} q'$ with $p'\ R\ q'$.
- For all $b \in B$, if $q \xrightarrow{b} q'$ then there exists some $p \xrightarrow{b} p'$ with $p'\ R\ q'$.

We write $p \lesssim_B q$ if $p\ R\ q$ for some partial bisimulation with bisimulation set B.

It is easy to see that partial bisimulation with bisimulation set B is a particular case of covariant-contravariant simulation.

Proposition 17. *Let P be an LTS. A relation R is a partial bisimulation with bisimulation set B iff it is a covariant-contravariant simulation for the same LTS when it is seen as a covariant-contravariant LTS with signature $A^r = A \setminus B$, $A^l = \emptyset$ and $A^{bi} = B$. As a consequence we have $p \lesssim_B q$ iff $p \lesssim_{cc} q$, for each $p, q \in P$.*

Proof. Immediate from the definitions. □

Remark 6. Note that, in the light of the discussion in Section 5, after having changed the signature of the LTS P in the manner described in the statement of the above result, checking whether $p \lesssim_B q$ holds in P can always be reduced to verifying whether $p \lesssim_{cc} q$ holds in $\mathcal{C}(\mathcal{M}(P))$. This check does *not* involve any bivariant action.

As a corollary of the above proposition, we immediately obtain the following result, that indicates us that, instead of the modal logic used in [4] to characterize the partial bisimulation preorder with bisimulation set B, one can use the simpler, negation-free logic for the covariant-contravariant simulation preorder.

Corollary 18. *Let p, q be states in some image-finite LTS. Then $p \lesssim_B q$ iff the collection of formulae in Definition 5 over the signature $A^r = A \setminus B$, $A^l = \emptyset$ and $A^{bi} = B$ satisfied by p is included in the collection of formulae satisfied by q.*

Note also that, as a corollary of Proposition 17, the translations of LTSs and formulae defined in Section 3 can be applied to embed LTSs modulo the partial bisimulation preorder into modal transition systems modulo refinement. In this case, however, there is an easier alternative transformation that does not require the extra state u.

Definition 16. Let P be an LTS over a set of actions A with a bisimulation set $B \subseteq A$. Then the MTS $\mathcal{N}(P)$ is constructed as follows:

- The set of states is that of P.
- For each transition $p \xrightarrow{a} p'$ in P, we add a transition $p \xrightarrow{a}_{\diamond} p'$ in $\mathcal{N}(P)$.
- For each transition $p \xrightarrow{b} p'$ in P with $b \in B$, we add a transition $p \xrightarrow{b}_{\Box} p'$ in $\mathcal{N}(P)$.
- There are no other transitions in $\mathcal{N}(P)$.

Proposition 19. *R is a partial bisimulation with bisimulation set B between P and Q iff R^{-1} is a refinement between $\mathcal{N}(Q)$ and $\mathcal{N}(P)$.*

Proof. ($\Rightarrow$) Assume that R is a partial bisimulation with bisimulation set B and suppose that $q\ R^{-1}\ p$. If $p \xrightarrow{a}_{\diamond} p'$ in $\mathcal{N}(P)$ then $p \xrightarrow{a} p'$ in P. Since R is a partial bisimulation, there is some $q \xrightarrow{a} q'$ in Q with $p'\ R\ q'$ and, by construction, $q \xrightarrow{a}_{\diamond} q'$ in $\mathcal{N}(Q)$ with $q'\ R^{-1}\ p'$. Now, if $q \xrightarrow{a}_{\Box} q'$ in $\mathcal{N}(Q)$ then $q \xrightarrow{a} q'$ in Q with $a \in B$. Since R is a partial bisimulation and $p\ R\ q$, there is some $p \xrightarrow{a} p'$ in P with $p'\ R\ q'$ and hence $p \xrightarrow{a}_{\Box} p'$ in $\mathcal{N}(P)$, as required.

($\Leftarrow$) Analogous. □

Remark 7. In the special case $B = \emptyset$, the partial bisimulation preorder is just the standard simulation preorder. Therefore, for the LTS defined by the term 0 we have, $0 \lesssim_B p$ for each state p in any LTS P. Since $B = \emptyset$, all the modal transition systems $\mathcal{N}(P)$ that result from the translation of an LTS P will have no must transitions; for such modal transition systems, $\mathcal{N}(P) \sqsubseteq 0$ always holds. Indeed, in that case $\sqsubseteq$ coincides with the inverse of the simulation preorder over MTSs.

The drawback of the direct transformation presented in Definition 16, as compared to that in Section 3, is that it does not preserve the satisfiability of modal formulae. The problem lies in the fact that, while the existential modality $\langle a \rangle$ allows any transition with $a \in A$ in the partial bisimulation framework, it requires a must transition in the setting of MTSs.

As we have seen, it is easy to express partial bisimulations as a special case of covariant-contravariant simulations. It is therefore natural to wonder whether the converse also holds. We shall present some indications that the partial bisimulation framework is strictly less expressive than both modal refinements and covariant-contravariant simulations.

Let us assume, by way of example, that the set of actions A is partitioned into $A^r = \{a\}$ and $A^l = \{b\}$—so the set of bivariant actions is empty. In this setting, there cannot be a translation $\mathcal{T}$ from LTSs modulo $\lesssim_{cc}$ into LTSs modulo $\lesssim_B$ that satisfies the following natural conditions (by abuse of notation, we identify an LTS P with a specific state p):

1. For all p and q, $p \lesssim_{cc} q \iff \mathcal{T}(p) \lesssim_B \mathcal{T}(q)$.
2. $\mathcal{T}$ is a homomorphism with respect to $+$, that is, $\mathcal{T}(p+q) = \mathcal{T}(p) + T(q)$, where $+$ denotes the standard notion of nondeterministic composition of LTSs from CCS [18]. (Intuitively, this compositionality requirement states that the translation only uses 'local information'.)
3. There is an n such that $\mathcal{T}(b^n)$ is not simulation equivalent to $\mathcal{T}(0)$, where b^n denotes an LTS consisting of n consecutive b-labelled transitions.

Indeed, observe that, by condition 2,

$$\mathcal{T}(p) = \mathcal{T}(p+0) = \mathcal{T}(p) + \mathcal{T}(0) \quad \text{for each } p,$$

and therefore $\mathcal{T}(p) + \mathcal{T}(0) \lesssim_B \mathcal{T}(p)$. This means that $\mathcal{T}(0) \lesssim \mathcal{T}(p)$ for each p, where $\lesssim$ is the simulation preorder. In particular, $\mathcal{T}(0) \lesssim \mathcal{T}(\bot)$ where $\bot$ is the process consisting of a b-labelled loop with one state, which is the least element with respect to $\lesssim_{cc}$.

Note now that $\bot \lesssim_{cc} b^{n+1} \lesssim_{cc} b^n \lesssim_{cc} 0$ for each $n > 0$. Therefore, by condition 1,

$$\mathcal{T}(\bot) \lesssim_B \mathcal{T}(b^{n+1}) \lesssim_B \mathcal{T}(b^n) \lesssim_B \mathcal{T}(0) \quad \text{for each } n > 0.$$

Hence,

$$\mathcal{T}(\bot) \lesssim \mathcal{T}(b^n) \lesssim \mathcal{T}(0) \lesssim \mathcal{T}(\bot) \quad \text{for each } n > 0.$$

This yields that, for each $n > 0$, $\mathcal{T}(b^n)$ is simulation equivalent to $\mathcal{T}(0)$, which contradicts condition 3. (Note that we have only used the soundness of the transformation $\mathcal{T}$.)

This is clearly indicating that any $\mathcal{T}$ that is compositional with respect to $+$ and is sound, in the sense of condition 1, would have to be very odd indeed, if it exists at all. Modulo simulation equivalence, such a translation would have to conflate a non-well-founded descending chain of LTSs into a single point, modulo simulation equivalence.

We end this section with a companion result.

Proposition 20. *Assume that $a \in A^r$ and $b \in A^l$. Suppose furthermore that $B = \emptyset$. Then there is no translation $\mathcal{T}$ from LTSs modulo $\lesssim_{cc}$ into LTSs modulo $\lesssim_B$ that satisfies conditions 1 and 2 above.*

Proof. Assume, towards a contradiction, that $\mathcal{T}$ is a translation from LTSs modulo $\lesssim_{cc}$ into LTSs modulo $\lesssim_B$ that satisfies the conditions in the statement of the proposition. Recall that, when B is empty, $\lesssim_B$ is the simulation preorder (see Remark 7). Therefore, using condition 2, for each p and q, we have that

$$\mathcal{T}(p) \lesssim_B \mathcal{T}(p) + \mathcal{T}(q) = \mathcal{T}(p + q).$$

This means, in particular, that $\mathcal{T}(a) \lesssim_B \mathcal{T}(a + b)$. By condition 1, it follows that $a \lesssim_{cc} a + b$. This is, however, false since b is in A^l. Therefore $\mathcal{T}$ cannot exist. □

8. Institutions and institution morphisms

In order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, we will now study their connections at a more abstract level, in the context of institutions [12]. When compared at the level of institutions it turns out that the correspondence between these models is, in a sense, not one-to-one.

Definition 17. The institution $\mathcal{I}_{cc} = (\mathbf{Sign}_{cc}, sen_{cc}, \mathbf{Mod}_{cc}, \models_{cc})$, associated with the logic for the covariant-contravariant simulation preorder, is defined as follows.

- $\mathbf{Sign}_{cc}$ has as objects triples (A, B, C) of pairwise disjoint sets and morphisms $f : A \cup B \cup C \longrightarrow A' \cup B' \cup C'$ with $f(A) \subseteq A'$, $f(B) \subseteq B'$, and $f(C) \subseteq C'$.
- $sen_{cc}(A, B, C)$ is the set of formulae in the logic characterizing the covariant-contravariant simulation preorder, with A the set of covariant actions, B the set of contravariant actions, and C the set of bivariant actions. For each signature morphism f and formula φ, the formula $sen(f)(\varphi)$ is obtained from φ by replacing each action a with $f(a)$.
- $\mathbf{Mod}_{cc}(A, B, C)$ is the category of LTSs over the set of actions $A \cup B \cup C$, with a distinguished state; a morphism from (P, p) to (Q, q) is a covariant-contravariant simulation R such that $(p, q) \in R$.

  Now, if $f : A \cup B \cup C \longrightarrow A' \cup B' \cup C'$ is a signature morphism, then

  $$\mathbf{Mod}_{cc}(f) : \mathbf{Mod}_{cc}(A', B', C') \longrightarrow \mathbf{Mod}_{cc}(A, B, C)$$

  maps P to $P|_f$ and $R : P \longrightarrow Q$ to $R_f : P|_f \longrightarrow Q|_f$, where:

  - The set of states of $P|_f$ is the same as that of P, and the distinguished state remains the same.

– $p \xrightarrow{a} p'$ in $P|_f$ if $p \xrightarrow{f(a)} p'$ in P.

– $R|_f$ coincides with R.

- $(P, s) \models_{cc} \varphi$ if $(P, s) \models \varphi$ using the notion of satisfaction associated with the logic for the covariant-contravariant simulation preorder given in Definition 5.

Proposition 21. *$\mathcal{I}_{cc}$ is an institution.*

Proof. It is a routine exercise to check that all defined notions are indeed categories and functors. As for the satisfaction condition, if $f : A \cup B \cup C \longrightarrow A' \cup B' \cup C'$ in $\mathbf{Sign}_{cc}$, $(P', s) \in \mathbf{Mod}_{cc}(A', B', C')$, and $\varphi \in sen_{cc}(A, B, C)$, then

$$(P', s) \models_{cc} sen_{cc}(f)(\varphi) \iff \mathbf{Mod}_{cc}(f)(P', s) \models_{cc} \varphi$$

can be proved by structural induction on φ. We consider the possible forms φ may have.

- $\top$ and $\bot$ are trivial.
- For $\varphi_1 \wedge \varphi_2$:

$$\begin{array}{lll} (P', s) \models_{cc} sen_{cc}(f)(\varphi_1 \wedge \varphi_2) & \iff & (P', s) \models_{cc} sen_{cc}(f)(\varphi_1) \wedge sen_{cc}(f)(\varphi_2) \\ & \overset{\text{IH}}{\iff} & (P'|_f, s) \models_{cc} \varphi_1 \text{ and } (P'|_f, s) \models_{cc} \varphi_2 \\ & \iff & (P'|_f, s) \models_{cc} \varphi_1 \wedge \varphi_2. \end{array}$$

- Analogously for $\varphi_1 \vee \varphi_2$.
- For $\langle a \rangle \varphi$, with $a \in A \cup C$:

$$\begin{array}{ll} (P', s) \models_{cc} sen_{cc}(f)(\langle a \rangle \varphi) & \\ \iff & (P', s) \models_{cc} \langle f(a) \rangle sen_{cc}(f)(\varphi) \\ \iff & \text{there is } s \xrightarrow{f(a)} p \text{ in } P' \text{ with } (P', p) \models_{cc} sen_{cc}(f)(\varphi) \\ \overset{\text{def } P'|_f, \text{ IH}}{\iff} & \text{there is } s \xrightarrow{a} p \text{ in } P'|_f \text{ with } (P'|_f, p) \models_{cc} \varphi \\ \iff & (P'|_f, s) \models_{cc} \langle a \rangle \varphi. \end{array}$$

- For $[a]\varphi$, with $a \in B \cup C$:

$$\begin{array}{ll} (P', s) \models_{cc} sen_{cc}(f)([a]\varphi) & \\ \iff & (P', s) \models_{cc} [f(a)] sen_{cc}(f)(\varphi) \\ \iff & (P', p) \models_{cc} sen_{cc}(f)(\varphi) \text{ for all } s \xrightarrow{f(a)} p \text{ in } P' \\ \overset{\text{def } P'|_f, \text{ IH}}{\iff} & (P'|_f, p) \models_{cc} \varphi \text{ for all } s \xrightarrow{a} p \text{ in } P'|_f \\ \iff & (P'|_f, s) \models_{cc} [a]\varphi. \end{array}$$

This completes the proof. □

Definition 18. The institution $\mathcal{I}_{mts} = (\mathbf{Sign}_{mts}, sen_{mts}, \mathbf{Mod}_{mts}, \models_{mts})$, associated with the logic for refinement over modal transition systems, is defined as follows.

- $\mathbf{Sign}_{mts}$ is the category of sets.
- $sen_{mts}(A)$ is the set of formulae over A in the logic presented in Definition 3. The formula $sen_{mts}(f)(\varphi)$ is obtained from φ by replacing each action a with $f(a)$.
- $\mathbf{Mod}_{mts}(A)$ is the category of MTSs over the set of labels A, with a distinguished state. A morphism from (M, m) to (N, n) is a refinement R such that $(m, n) \in R$.

  If $f : A \longrightarrow B$ in $\mathbf{Sign}_{mts}$, then $\mathbf{Mod}_{mts}(f) : \mathbf{Mod}_{mts}(B) \longrightarrow \mathbf{Mod}_{mts}(A)$ maps an MTS M to $M|_f$ and a morphism R to $R|_f$, where:

  - $M|_f$ has the same set of states as M and the same distinguished state.
  - $p \xrightarrow{a}_\diamond p'$ in $M|_f$ if $p \xrightarrow{f(a)}_\diamond p'$ in M.
  - $p \xrightarrow{a}_\Box p'$ in $M|_f$ if $p \xrightarrow{f(a)}_\Box p'$ in M.
  - $R|_f$ coincides with R.
- $\models_{mts}$ is the notion of satisfaction presented in Definition 3.

Proposition 22. *$\mathcal{I}_{mts}$ is an institution.*

Proof. Again, let us just prove the satisfaction condition

$$(M', s) \models_{mts} sen_{mts}(f)(\varphi) \iff \mathbf{Mod}_{mts}(f)(M', s) \models_{mts} \varphi,$$

for $f : A \longrightarrow B$ in $\mathbf{Sign}_{mts}$, $(M', s) \in \mathbf{Mod}_{mts}(B)$, and $\varphi \in sen_{mts}(A)$, by induction on φ. We consider the possible forms φ may have.

- $\top$ and $\bot$ are trivial.
- For $\varphi_1 \wedge \varphi_2$:

$$\begin{aligned}
&(M', s) \models_{mts} sen_{mts}(f)(\varphi_1 \wedge \varphi_2) \\
\iff\quad &(M', s) \models_{mts} sen_{mts}(f)(\varphi_1) \wedge sen_{mts}(f)(\varphi_2) \\
\overset{\text{IH}}{\iff}\quad &(M'|_f, s) \models_{mts} \varphi_1 \text{ and } (M'|_f, s) \models_{mts} \varphi_2 \\
\iff\quad &(M'|_f, s) \models_{mts} \varphi_1 \wedge \varphi_2.
\end{aligned}$$

- Analogously for $\varphi_1 \vee \varphi_2$.

- For $\langle a\rangle\varphi$:

$$\begin{array}{rll}
(M', s) \models_{mts} sen_{mts}(f)(\langle a\rangle\varphi) & & \\
\iff & (M', s) \models_{mts} \langle f(a)\rangle sen_{mts}(f)(\varphi) & \\
\iff & \text{there is } s \overset{f(a)}{\rightarrow}_{\Box} p \text{ in } M' \text{ with } (M', p) \models_{mts} sen_{mts}(f)(\varphi) & \\
\overset{\text{def } M'|_f,\ \text{IH}}{\iff} & \text{there is } s \overset{a}{\rightarrow}_{\Box} p \text{ in } M'|_f \text{ with } (M'|_f, p) \models_{mts} \varphi & \\
\iff & (M'|_f, s) \models_{mts} \langle a\rangle\varphi. &
\end{array}$$

- For $[a]\varphi$:

$$\begin{array}{rll}
(M', s) \models_{mts} sen_{mts}(f)([a]\varphi) & & \\
\iff & (M', s) \models_{mts} [f(a)] sen_{mts}(f)(\varphi) & \\
\iff & (M', p) \models_{mts} sen_{mts}(f)(\varphi) \text{ for all } s \overset{f(a)}{\rightarrow}_{\Diamond} p \text{ in } M' & \\
\overset{\text{def } M'|_f,\ \text{IH}}{\iff} & (M'|_f, p) \models_{mts} \varphi \text{ for all } s \overset{a}{\rightarrow}_{\Diamond} p \text{ in } M'|_f & \\
\iff & (M'|_f, s) \models_{mts} [a]\varphi. & \Box
\end{array}$$

As the following result shows, one can translate $\mathcal{I}_{mts}$ into $\mathcal{I}_{cc}$ using an institution morphism [13]. (The intuition for institution morphisms is that they are truth preserving translations from one logical system into another.)

Proposition 23. *$(\Phi, \alpha, \beta) : \mathcal{I}_{mts} \longrightarrow \mathcal{I}_{cc}$ is an institution morphism, where:*

- *$\Phi : \mathbf{Sign}_{mts} \longrightarrow \mathbf{Sign}_{cc}$ maps A to the triple $(\mathrm{cv}(A), \mathrm{ct}(A), \emptyset)$, with:*
  - *$\mathrm{cv}(A) = \{\mathrm{cv}(a) \mid a \in A\}$ and*
  - *$\mathrm{ct}(A) = \{\mathrm{ct}(a) \mid a \in A\}$.*

  For $f : A \longrightarrow B$, we define $\Phi(f)(\mathrm{cv}(a)) = \mathrm{cv}(f(a))$ and $\Phi(f)(\mathrm{ct}(a)) = \mathrm{ct}(f(a))$.
- *The natural transformation $\alpha : sen_{cc} \circ \Phi \Rightarrow sen_{mts}$ translates a formula φ in $sen_{cc}(\mathrm{cv}(A), \mathrm{ct}(A), \emptyset)$ as follows:*
  - *$\alpha(\top) = \top$, $\alpha(\bot) = \bot$.*
  - *$\alpha(\varphi_1 \wedge \varphi_2) = \alpha(\varphi_1) \wedge \alpha(\varphi_2)$.*
  - *$\alpha(\varphi_1 \vee \varphi_2) = \alpha(\varphi_1) \vee \alpha(\varphi_2)$.*
  - *$\alpha(\langle \mathrm{cv}(a)\rangle\varphi) = \langle a\rangle\alpha(\varphi)$.*
  - *$\alpha([\mathrm{ct}(a)]\varphi) = [a]\alpha(\varphi)$.*
- *The natural transformation $\beta : \mathbf{Mod}_{mts} \Rightarrow \mathbf{Mod}_{cc} \circ \Phi$ maps an MTS (M, s) in $\mathbf{Mod}_{mts}(A)$ to $(\mathcal{C}(M), s)$, and a morphism R to itself.*

PROOF. For A in $\mathbf{Sign}_{mts}$, (M,s) in $\mathbf{Mod}_{mts}(A)$, and φ in $sen_{cc}(\Phi(A))$, we prove the satisfaction condition

$$(M,s) \models_{mts} \alpha(\varphi) \iff \beta(M,s) \models_{cc} \varphi$$

by induction on φ. The only non-trivial cases correspond to formulae of the form $\langle \mathrm{cv}(a)\rangle\varphi$ and $[\mathrm{ct}(a)]\varphi$.

- For $\langle \mathrm{cv}(a)\rangle\varphi$, we reason thus:

$$\begin{array}{lcl}
(M,s) \models \alpha(\langle \mathrm{cv}(a)\rangle\varphi) & \iff & (M,s) \models \langle a\rangle\alpha(\varphi) \\
& \iff & \text{there is } s \xrightarrow{a}_{\Box} p \text{ in } M \text{ with } (M,p) \models \alpha(\varphi) \\
& \overset{\text{IH}}{\iff} & \text{there is } s \overset{\mathrm{cv}(a)}{\to} p \text{ in } \mathcal{C}(M) \text{ with } (\mathcal{C}(M),p) \models \varphi \\
& \iff & (\mathcal{C}(M),s) \models \langle \mathrm{cv}(a)\rangle\varphi.
\end{array}$$

- For $[\mathrm{ct}(a)]\varphi$, we argue as follows:

$$\begin{array}{lcl}
(M,s) \models \alpha([\mathrm{ct}(a)]\varphi) & \iff & (M,s) \models [a]\alpha(\varphi) \\
& \iff & (M,p) \models \alpha(\varphi) \text{ for all } s \xrightarrow{a}_{\diamond} p \text{ in } M \\
& \overset{\text{IH}}{\iff} & (\mathcal{C}(M),p) \models \varphi \text{ for all } s \overset{\mathrm{ct}(a)}{\to} p \text{ in } \mathcal{C}(M) \\
& \iff & (\mathcal{C}(M),s) \models [\mathrm{ct}(a)]\varphi.
\end{array}$$

  This completes the proof. □

The import of the above result is that MTSs modulo refinement and their accompanying modal logic can be 'translated in a truth preserving fashion' into LTSs modulo the covariant-contravariant simulation preorder and their companion modal logic. It is natural to ask oneself whether one can consider $\mathcal{I}_{mts}$ a 'subinstitution' of $\mathcal{I}_{cc}$. There are several related notions of subinstitution that have in common the requirement that the functor β, which is used to translate the models between the institutions, is an equivalence of categories.

Recall that an object in a category is *weakly final* if any other object has at least one arrow into it.

Proposition 24. $\mathbf{Mod}_{cc}(A,B,\emptyset)$ *has weakly final objects but* $\mathbf{Mod}_{mts}(A)$ *does not.*

PROOF. First, consider the pair (F,s) where F is the LTS with a single state s and transitions $s \xrightarrow{a} s$ for every $a \in A$. (Note that, if A is empty, then (F,s) is just the LTS 0.) It is immediate to check that (F,s) is a weakly final object of $\mathbf{Mod}_{cc}(A,B,\emptyset)$.

Now, assume that (F',s') is weakly final in $\mathbf{Mod}_{mts}(A)$ and consider the following two MTSs:

- (M,m), with m the only state in M and transitions $m \xrightarrow{a}_{\Box} m$ (and $m \xrightarrow{a}_{\diamond} m$) for every $a \in A$.

- (N, n), with n the only state in N and no transitions.

The existence of a morphism, that is a refinement, from (M, m) to (F', s') implies that, for every $a \in A$, there must be transitions of the form $s' \xrightarrow{a}_{\Box} s'_a$ in F' for some s'_a; therefore, there are also transitions $s \xrightarrow{a}_{\diamond} s'_a$. But then, the morphism from (N, n) to (F', s') requires the existence of transitions $n \xrightarrow{a}_{\diamond} n$ in N, which do not exist by the definition of N. Hence, there is no weakly final object in $\mathbf{Mod}_{mts}(A)$. □

In other words, in the absence of bivariant actions, there is a universal implementation in the setting of LTSs modulo the covariant-contravariant simulation preorder. Within that framework, there is also a universal specification, namely the LTS (I, s) where I is the LTS with a single state s and transitions $s \xrightarrow{b} s$ for every $b \in B$. On the other hand, there is a universal specification with respect to modal refinements, namely the MTS U from Example 1, but no universal implementation.

Proposition 25. *There cannot exist an embedding (Φ, α, β) from $\mathcal{I}_{mts}$ into $\mathcal{I}_{cc}$ such that $\Phi(A)$ does not have bivariant actions for some A.*

Proof. If such an embedding existed then β_A, which is the natural transformation translating MTSs into LTSs and refinement relations into covariant-contravariant simulations, would be an equivalence between $\mathbf{Mod}_{mts}(A)$ and $\mathbf{Mod}_{cc}(\Phi(A))$. Since equivalences of categories preserve weakly final objects, the result follows from Proposition 24. □

We will now argue that $\mathcal{I}_{mts}$ cannot be embedded into $\mathcal{I}_{cc}$ even in the presence of bivariant actions. Recall that an object in a category is *weakly initial* if there is at least one arrow from it into any other object.

Proposition 26. $\mathbf{Mod}_{mts}(A)$ *has weakly initial objects but* $\mathbf{Mod}_{cc}(A, B, C)$ *does not if $C \neq \emptyset$.*

Proof. Consider the MTS (I, s) defined by $s \xrightarrow{a}_{\diamond} s$ for all $a \in A$. We have already seen that it is weakly initial.

Now, assume that (I', s') is weakly initial in $\mathbf{Mod}_{cc}(A, B, C)$ and let $c \in C$. We define the following LTSs:

- (P, p) with $p \xrightarrow{c} p$, and
- (Q, q) with a single state q and no transitions.

A morphism from (I', s') to (P, p) requires a transition $s' \xrightarrow{c} s''$ in I' for some s''. But then, a morphism from (I', s') to (Q, q) requires a transition $q \xrightarrow{c} q$, which does not exist by definition. Therefore, (I', s') cannot exist. □

Proposition 27. *There cannot exist an embedding (Φ, α, β) from $\mathcal{I}_{mts}$ into $\mathcal{I}_{cc}$ such that $\Phi(A)$ has bivariant actions for some A.*

PROOF. Such an embedding β_A, if it existed, would be an equivalence of categories between $\mathbf{Mod}_{mts}(A)$ and $\mathbf{Mod}_{cc}(\Phi(A))$. This cannot hold by Proposition 26 because equivalences of categories preserve weakly initial objects. □

A natural question to ask is whether there is an embedding from $\mathcal{I}_{cc}$ into $\mathcal{I}_{mts}$. The following proposition answers this question negatively.

Proposition 28. *There exists no embedding from $\mathcal{I}_{cc}$ into $\mathcal{I}_{mts}$.*

PROOF. If such an embedding (Φ, α, β) existed, $\beta_{(A,B,\emptyset)}$ would be an equivalence between $\mathbf{Mod}_{cc}(A, B, \emptyset)$ and $\mathbf{Mod}_{mts}(\Phi(A, B, \emptyset))$, which is not possible by Proposition 24 because equivalences preserve weakly final objects. □

In [1] we conjectured that there is not even an institution morphism from $\mathcal{I}_{cc}$ to $\mathcal{I}_{mts}$; we now make this claim precise.

If we are not concerned about how contrived this morphism can be, then a trivial one can indeed be defined. Let Φ map any signature to the singleton set $\{1\}$, β map any LTS to a MTS with a single state s and transitions $s \xrightarrow{1}_{\diamond} s$ and $s \xrightarrow{1}_{\Box} s$, and α be recursively defined by $\alpha([1]\varphi) = \alpha(\varphi)$, $\alpha(\langle 1\rangle\varphi) = \alpha(\varphi)$, and as expected in the remaining cases. It is then a simple exercise to check that (Φ, α, β) satisfies the conditions to be an institution morphism, however trivial and artificial it may be.

Taking Proposition 23 as a model, and recalling the good properties of the function $\mathcal{M}$ studied in Section 3, a "natural" morphism from $\mathcal{I}_{cc}$ to $\mathcal{I}_{mts}$ would be expected to satisfy $\beta(M, s) = (\mathcal{M}(M), s)$. We now argue that such a morphism cannot exist.

Assume that $(A, B, C) \in \mathbf{Sign}_{cc}$, let $a \in A$ be any covariant action, and $[a]\bot$ a Boudol-Larsen modal formula: how should $\alpha([a]\bot)$ be defined? By the requirements of institution morphisms, the following equivalence must hold for all LTS M:

$$(M, s) \models_{cc} \alpha([a]\bot) \iff \beta(M, s) \models_{mts} [a]\bot.$$

The right-hand side is true iff $(\mathcal{M}(M), s') \models_{mts} \bot$ for all $s \xrightarrow{a}_{\diamond} s'$ in $\mathcal{M}(M)$ which, by construction, only holds if there is no s' in M with $s \xrightarrow{a} s'$ in M. Therefore, $\alpha([a]\bot)$ has to be such that:

- $(M, s) \models_{cc} \alpha([a]\bot)$ if there is no $s \xrightarrow{a} s'$ in M, but
- $(M, s) \not\models_{cc} \alpha([a]\bot)$ if there is $s \xrightarrow{a} s'$ in M.

The immediate candidate would be $[a]\bot$ itself, now considered as a covariant-contravariant modal formula, but this is not possible since in this framework the modality $[\_]$ requires a contravariant action. Actually, no such formula can be defined which means that no institution morphism with $\beta(M, s) = (\mathcal{M}(M), s)$ can exist.

Note also that, although not necessary for our negative results above, we could have associated a more general institution $\mathcal{I}'_{cc}$ to the logic for the covariant-contravariant simulation preorder as follows:

- $\mathbf{Sign}'_{cc}$ has as objects triples (A, B, C) of pairwise disjoint sets and morphisms are relations $R \subseteq (A \times A') \cup (B \times B') \cup (C \times C')$.

- $sen'_{cc}(A, B, C)$ is the set of formulae in the logic characterizing the covariant-contravariant simulation preorder, with A the set of covariant actions, B the set of contravariant actions, and C the set of bivariant actions. For each morphism R and formula φ, the formula $sen'_{cc}(R)(\varphi)$ is obtained from φ by "replacing" each action a with every a' such that aRa'. More precisely, $sen'_{cc}(R)(\varphi)$ is defined recursively so that $\langle a\rangle\varphi'$ becomes $\bigvee_{aRa'}\langle a'\rangle sen'_{cc}(R)(\varphi')$ and $[b]\varphi'$ becomes $\bigwedge_{bRb'}[b']sen'_{cc}(R)(\varphi')$.

- $\mathbf{Mod}'_{cc}(A, B, C)$ is the category of LTSs over the set of actions $A \cup B \cup C$, with a distinguished state; a morphism from (P, p) to (Q, q) is a covariant-contravariant simulation S such that $(p, q) \in S$.

  Now, if $R : (A, B, C) \longrightarrow (A', B', C')$ is a $\mathbf{Sign}'_{cc}$-signature morphism, then

  $$\mathbf{Mod}'_{cc}(R) : \mathbf{Mod}'_{cc}(A', B', C') \longrightarrow \mathbf{Mod}'_{cc}(A, B, C)$$

  maps P to $R(P)$ and a simulation $S : P \longrightarrow Q$ to $R(S) : R(P) \longrightarrow R(Q)$, where:

  - The set of states of $R(P)$ is the same as that of P, and the distinguished state remains the same.
  - $p \xrightarrow{a} p'$ in $R(P)$ if aRa' and $p \xrightarrow{a'} p'$ in P.
  - $R(S)$ coincides with S.

- $(P, s) \models'_{cc} \varphi$ if $(P, s) \models \varphi$ using the notion of satisfaction associated with the logic for the covariant-contravariant simulation preorder given in Definition 5.

That is, signature morphisms become arbitrary relations that "preserve" the modality of the actions.

Obviously, the institution $\mathcal{I}_{mts}$ could be subjected to an analogous generalization; then, it would be a simple exercise to translate to this new setting the results proved in Propositions 21–28.

9. Conclusions and future work

In this paper we have studied the relationships between three notions of behavioural preorders that have been proposed in the literature: refinement over modal transition systems, and the covariant-contravariant simulation and the partial bisimulation preorders over labelled transition systems. We have provided mutual translations between modal transition systems and labelled transition systems that preserve, and reflect, refinement and the covariant-contravariant simulation preorder, as well as the the modal properties that can be expressed in the logics that characterize those preorders. We have also offered a translation from labelled transition systems modulo the partial bisimulation

preorder into the same model modulo the covariant-contravariant simulation preorder, together with some evidence that the former model is less expressive than the latter. Finally, in order to gain more insight into the relationships between modal transition systems modulo refinement and labelled transition systems modulo the covariant-contravariant simulation preorder, we have also phrased and studied their connections in the context of institutions.

The work presented in the study opens several interesting avenues for future research. Here we limit ourselves to mentioning a few research directions that we plan to pursue in future work.

First of all, it would be interesting to study the relationships between the LTS-based models we have considered in this article and variations on the MTS model surveyed in, for instance, [3]. In particular, the third author recently contributed in [10] to the comparison of several refinement settings, including modal and mixed transition systems. The developments in that paper offer a different approach to the comparison and application of the formalisms studied in this article.

In [9], three of the authors gave a ground-complete axiomatization of the covariant-contravariant simulation preorder over the language BCCS [18]. It would be interesting to see whether the translations between MTSs and LTSs we have provided in this paper can be used to lift that axiomatization result, as well as results on the nonexistence of finite (in)equational axiomatizations, to the setting of modal transition systems modulo refinement, using the BCCS-like syntax for MTSs given in [6] and used in Section 6 of this paper. We also intend to study whether our translations can be used to obtain characteristic-formula constructions [6, 14, 21] for one model from extant results on the existence of characteristic formulae for the other. In the setting of the finite LTSs that are the image of MTS terms via $\mathcal{C}$, this has been achieved in Section 6 of this study.

The existence of characteristic formulae allows one to reduce checking the existence of a behavioural relation between two processes to a model checking question. Conversely, the main result from [6] offers a complete characterization of the model checking questions of the form $(M, m) \models \varphi$, where M is an MTS and φ is a formula in the logic for MTSs considered in this paper, that can be reduced to checking for the existence of a refinement between (M_φ, m_φ) and (M, m), where (M_φ, m_φ) is an MTS with a distinguished state that 'graphically represents' the formula φ. In [2], we offered a characterization of the logical specifications that can be 'graphically represented' by LTSs modulo the covariant-contravariant simulation preorder and partial bisimilarity. This result applies directly to LTSs whose signature contains no bivariant actions. Such a characterization may shed further light on the relative expressive power of the two formalisms and may give further evidence of the fact that LTSs modulo the covariant-contravariant simulation preorder are, in some suitable formal sense, more expressive than LTSs modulo partial bisimilarity.

Last, but not least, the development of the notion of partial bisimulation in [4, 5] has been motivated by the desire to develop a process-algebraic model within which one can study topics in the field of *supervisory control* [20]. Recently, MTSs have been used as a suitable model for the specification of service-

oriented applications, and results on the supervisory control of systems whose specification is given in that formalism have been presented in, e.g., [7, 11]. It is a very interesting area for future research to study whether the mutual translations between MTSs modulo refinement and LTSs modulo the covariant-contravariant simulation preorder can be used to transfer results on supervisory control from MTSs to LTSs. One may also wish to investigate directly the adaptation of the supervisory control theory of Ramadge and Wonham to the enforcement of specifications given in terms of LTSs modulo the covariant-contravariant simulation preorder.

[1] L. Aceto, I. Fábregas, D. de Frutos Escrig, A. Ingólfsdóttir, and M. Palomino. Relating modal refinements, covariant-contravariant simulations and partial bisimulations. In F. Arbab and M. Sirjani, editors, *Fundamentals of Software Engineering, FSEN 2011*, Lecture Notes in Computer Science. Springer, 2011. To appear.

[2] L. Aceto, I. Fábregas, D. de Frutos Escrig, A. Ingólfsdóttir, and M. Palomino. Graphical representation of covariant-contravariant modal formulae. In B. Luttik and F. Valencia, editors, *Proceedings of EXPRESS 2011, the 18th International Workshop on Expressiveness in Concurrency*, volume 64 of Electronic Proceedings in Theoretical Computer Science, pages 1–15, 2011.

[3] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wąsowski. 20 years of modal and mixed specifications. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 95:94–129, 2008.

[4] J. Baeten, D. van Beek, B. Luttik, J. Markovski, and J. Rooda. Partial bisimulation. SE Report 2010-04, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology, 2010.

[5] J. Baeten, D. van Beek, A. van Hulst, and J. Markovski. A process algebra for supervisory coordination. In L. Aceto and M. Mousavi, editors, *Proceedings First International Workshop on Process Algebra and Coordination, PACO 2011*, volume 60 of Electronic Proceedings in Theoretical Computer Science, pages 36–55, 2011.

[6] G. Boudol and K. G. Larsen. Graphical versus logical specifications. *Theoretical Comput. Sci.*, 106(1):3–20, 1992.

[7] P. Darondeau, J. Dubreil, and H. Marchand. Supervisory control for modal specifications of services. In *Proceedings of WODES 2010, August 30–September 1, 2010, Berlin, Germany*, 2010. To appear.

[8] I. Fábregas, D. de Frutos-Escrig, and M. Palomino. Non-strongly stable orders also define interesting simulation relations. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *Algebra and Coalgebra in Computer Science, Third*

conference CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings, volume 5728 of *Lecture Notes in Computer Science*, pages 221–235. Springer-Verlag, 2009.

[9] I. Fábregas, D. de Frutos-Escrig, and M. Palomino. Logics for contravariant simulations. In J. Hatcliff and E. Zucca, editors, *Formal Techniques for Distributed Systems, Joint 12th IFIP WG 6.1 International Conference, FMOODS 2010 and 30th IFIP WG 6.1 International Conference, FORTE 2010, Amsterdam, The Netherlands, June 7-9, 2010. Proceedings*, volume 6117 of *Lecture Notes in Computer Science*, pages 224–231. Springer-Verlag, 2010.

[10] H. Fecher, D. de Frutos-Escrig, G. Lüttgen, and H. Schmidt. On the expressiveness of refinement settings. In F. Arbab and M. Sirjani, editors, *Fundamentals of Software Engineering, Third IPM International Conference, FSEN 2009, Kish Island, Iran, April 15-17, 2009, Revised Selected Papers*, volume 5961 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, 2009.

[11] G. Feuillade and S. Pinchinat. Modal specifications for the control theory of discrete event systems. *Discrete Event Dynamical Systems*, 17:211–232, 2007.

[12] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.

[13] J. A. Goguen and G. Rosu, Institution morphisms. *Formal Aspects of Computing* 13:274–307, 2002.

[14] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68(1–3):125–145, Jan./Feb./Mar. 1986.

[15] J. Hughes and B. Jacobs. Simulations in coalgebra. *Theoretical Comput. Sci.*, 327(1–2):71–108, 2004.

[16] K. G. Larsen. Modal specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer-Verlag, 1989.

[17] K. G. Larsen and B. Thomsen. A modal process logic. In *Proceedings* 3^{th} *Annual Symposium on Logic in Computer Science,* Edinburgh, pages 203–210. IEEE Computer Society Press, 1988.

[18] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[19] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *5th GI Conference,* Karlsruhe, Germany, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[20] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25:206–230, 1987.

[21] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.